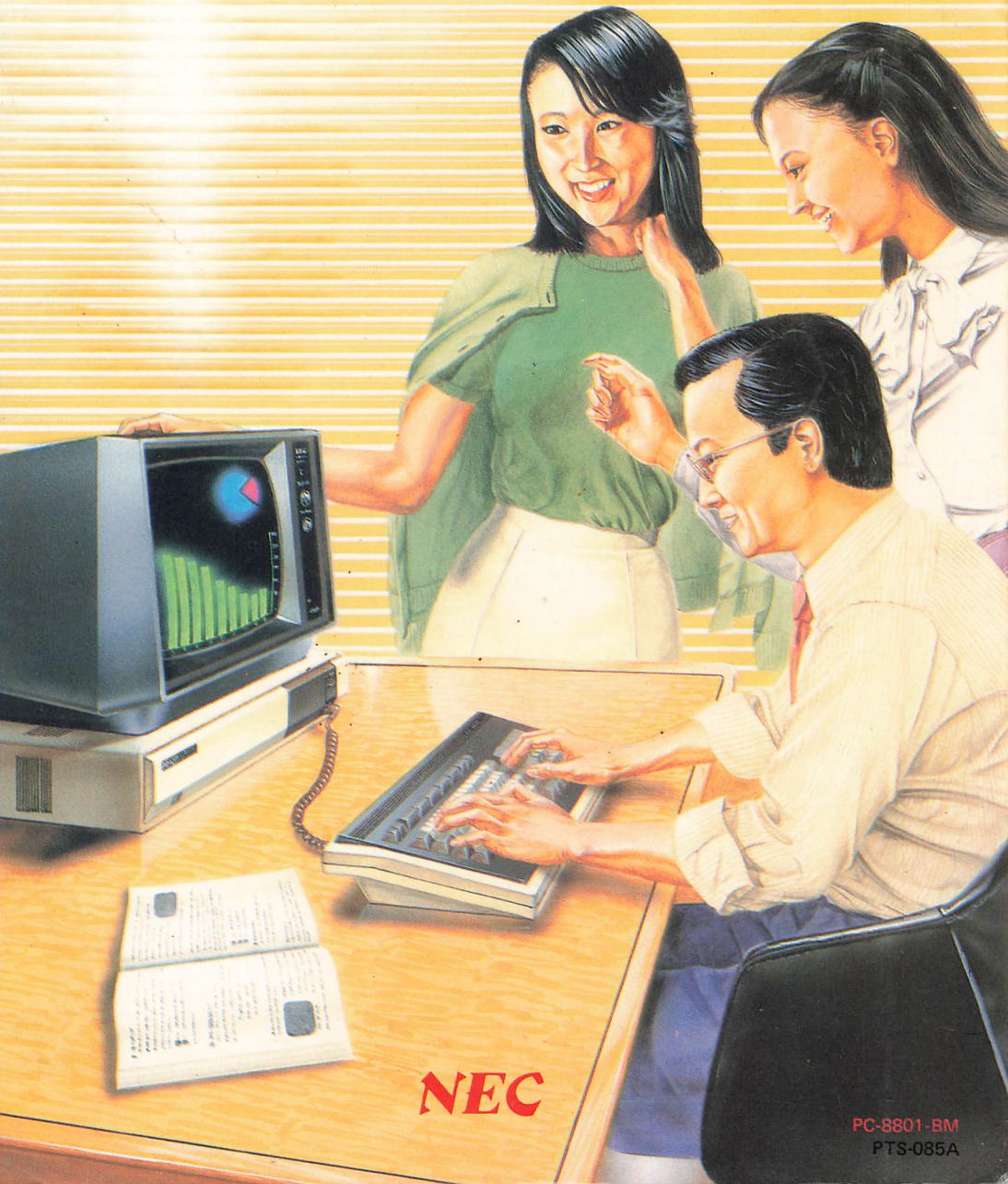
# PC-8801

## N88-BASIC入門



NEC

PC-8801-BM
PTS-085A

**御注意**

(1)本書の内容の一部又は全部を無断転載することは禁止されています。
(2)本書の内容に関しては将来予告なしに変更することがあります。
(3)本書は内容について万全を期して作成いたしましたが，万一御不審な点や誤り，記載もれなどお気付きのことがありましたら御連絡下さい。
(4)運用した結果の影響については(3)項にかかわらず責任を負いかねますので御了承下さい。

# N88-BASIC入門

# まえがき

本書の内容は，PC-8801のN88-BASICリファレンス・マニュアル，ユーザース・マニュアルとは，大きな違いがあります．

リファレンス・マニュアル，ユーザース・マニュアルが，PC-8801及びN88-BASICの基本的な機能について解説しているのに対して，本書はPC-8801のN88-BASICを基本に，初歩のプログラミングについて優しく解説した"入門書"です．ですから，PC-8801の機能については，ユーザース・マニュアルを，N88-BASICの各コマンド・ステートメントの働きについては，リファレンス・マニュアルをそれぞれ参照してください．

実際にPC-8801を使ってプログラムが組めるようになるには，リファレンス，ユーザース・マニュアルを読んだだけでは，どうしても無理があります．それは初心者にとって，まず第一に身に付けるべきプログラミングについての基本的な考え方が備わっていないと，理解しにくい点があるためです．

本書は，PC-8801を直接自分の手で操作しながら，PC-8801の特徴やプログラミングについての基本的な考えを勉強していきます．「習うより慣れろ」を主旨としていますので，実際にPC-8801を動かしながら勉強を進めていってください．PC-8801を使って，どんなことでも自分で確認しながら勉強をしていくことが，プログラミング技術の向上につながるのです．コンピュータはあなたが使うものであって，コンピュータ自身が何かをしてくれる訳ではありません．最後まであきらめずに，本書を読んでしっかり勉強してください．PC-8801の機能をフルに活用できるのも，押入れに入れてしまうのも，あなた次第です．

本書を読み終えた時，あなたはもう立派なプログラマーに変身していることでしょう．

# もくじ

まえがき……3

## 第1章　PC-8801の紹介

はじめに……11
これだけは最低限必要……11
電源を入れてみよう……12
キーボードを操作する……13
特殊キーにはこんなものが……25
目的のキーを探す……28
コンピュータとカセットレコーダ……30
サンプルプログラムを読む……31
サンプルプログラムを実行する……33
サンプルプログラムを止める……34
サンプルプログラムを見る……35
まとめ……35

## 第2章　BASICに慣れよう

"BASIC"って何？……39
ダイレクトモードで使う……40
文字の色を変えてみよう……42
画面をクリアする……44
↵(リターンキー)その働きは？……46
スクリーンエディタ　(Part I)……47
打ち間違いに気付いたら……48
PRINT命令を使う……50
電卓と比べると……51

A＝A＋1 !? （代入文）……55
まとめ……58
フロッピーディスク装置をお持ちのあなたへ……58
ディスケットは非常にデリケート……59
ロード，セーブの仕方……61
ディスケットのクラッシュに備える……63
まとめ……64

## 第3章　BASICプログラミングの初歩

“プログラム”って何？……67
“ソフトウェア”＝“プログラム” ??……68
今までとはここが違う……68
プログラミングのお話あれこれ……70
ロード・ろーど・LOAD……71
プログラムを見る……72
「走れ！」……74
そして何もなくなった……74
さあ，もう一度復習しましょう……75
Basic PROGRAMMING……78
プログラミングにおける主な登場人物……78
監督はもちろんあなた……87
実行はコンピュータに任せる……87
プログラムの保存は磁気の力で……88
カセットテープはこう使う……90
ピーヒョロロ……90
頭の痛くなるお話……90

IF～GOTO～のループについて……94
恐怖の永久ループにあなたも襲われる……96
ループ脱出計画……97
もっとループを……97
IF～GOTO～は万能選手……100
FORとNEXTでサンドイッチ？……101
FORの中にまたFOR……103
初歩のプログラミングヒント……104
:(コロン)と;(セミコロン),.(ピリオド)と,(コンマ)は同じようでも大違い……105
まとめ……106

## 第4章　グラフィックスがいっぱい

第3章まで読んでこられた方へ……109
とにかく描いてみる　(キャラクタ)……110
まず最初にすることは　(モードの設定)……111
画面をクリアする……112
点を打つ……114
点を消す……116
点をつなげると…　……120
もっと簡単に……120
ボックス(箱)を描く……121
円を描くには……122
塗りつぶす……123
絵が動く……125
ビジネスに使うと　(円グラフ，棒グラフ)……126
美しい図形を描くには……129
まとめ……131

## 第5章　ストリングと配列


あなたならどうする……135
テーブルの上に並べると…　……135
使わないと損をするDIM命令……136
同じ変数名，コンピュータはどう区別する……138
配列のエラーメッセージ，その傾向と対策……139
文字列(ストリング)を使う……140
なぜストリングなのか？……141
果してストリングはつながるか……144
ストリングは長さを持っている……144
キャラクタコード表を作る……145
CHR$とASCは表と裏……147
時間を切る……147
SPACE$はSPCを兼ねる？……149
便利な命令STRING$……150
似ているようでも全然違う……151
数値とストリングの変換はこの命令で……151
数値と比べると……152
ストリングをサーチする……153
まとめ……155


## APPENDIX


A.　スクリーンエディタ　PartII……159
B.　時計，日付プログラム……164
C.　キャラクタセット表……165
D.　PC-8001ソフトウェアを利用するには……166

# 第1章

# PPC-8801の紹介

## はじめに

これからあなたと勉強を共にするPC-8801は，その上で走る$N_{88}$-BASICと共にいままでにない強力な機能をもっています．たしかに，高解像度グラフィック，強力な命令群など，極端な話をすると，使いこなすのに骨がおれるとも言うほどです．

でも，心配することはありません．本書ではBASICとはなにかが全くわからない人のために，その基礎の基礎から読んでいけるよう，特に実際にPC-8801を使いながら読んでいけるように書かれています．気楽に読み進んでください．

BASIC言語にかぎらず，コンピュータを使いこなすということは，ちょうど外国語を話すことと似ています．どんどん練習して，理論よりも実際に慣れ親しんでください．よほど無茶苦茶なことをしないかぎり，どんな操作をしてもPC-8801は壊れることはないのですから．しかし，初めてBASICを動かす方にとっては，今は不安で一杯だと思います．中でも，あなたの目の前にある高価なPC-8801を壊してしまうのではないかと心配ではありませんか．それに知らない言葉がやたらと出てくるので，少々気遅れしているのではないでしょうか．知らない言葉がたくさん出てくると，確かに読みづらいですね．よくわかります．けれども，少しの間は我慢して読んでみてください．何も1回読んだだけで，すべてを理解しなければならないという訳ではないのです．逆に，そんなことをしていたのではすぐに疲れてしまいます．のんびりと，やっていきましょう．

しかし，1つだけあなたにお願いがあります．それは，本を読んだだけでわかったような気になって欲しくないということです．あなたがPC-8801をお持ちなら，必ず，どんなつまらない事でも，実際にPC-8801にやらせてみて欲しいのです．PC-8801だって，それを待っているのです．大いに使ってあげてください．

## これだけは最低限必要

さて，それではPC-8801を動かす準備に入りましょう．必要なものは，

PC-8801　本体及びキーボード
ディスプレイ装置
カセットテープレコーダ
接続ケーブル
付属の取扱い説明書及びマニュアル

これだけは，最低限必要です．ディスプレイ装置は，グリーン（白黒）ディスプ

レイ装置でもカラーディスプレイ装置でも構いませんが，一応，カラーディスプレイ装置を対象にして，これから先話を進めていきます．

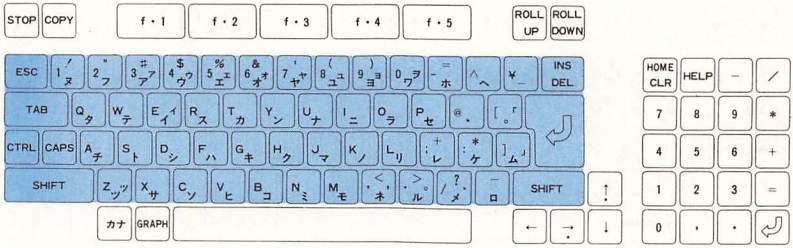
これらの装置を，PC-8801ユーザースマニアル（PC-8801接続説明図）にしたがって，間違えないように接続ケーブルでつないでいきます．大切な所ですから，慎重にどうぞ．PC-8801 を置く場所を初めから決めていれば，これらの接続は 1 回きりで後からはずすことはないでしょうが，もし，後から PC-8801 の置く場所を変えるために，接続ケーブルをはずして，場所を動かした後で，再び PC-8801 をつなぐ時に，接続の仕方がわからなくなったら，迷わずに取扱い説明書を見直しましょう．うろ覚えでの接続は，PC-8801 を壊してしまうかもしれません．もちろん，接続がすべて終わるまで，電源プラグをコンセントに差し込んではいけません．何度でも，自分が納得するまで，正しく接続されているか確認しておきましょう．もし，すべての接続が完了したのであれば，次へ進みましょう．

## 電源を入れてみよう

すべての装置が正しく接続できたなら，いよいよ，待望の瞬間が迫っています．まずは，それぞれの装置の電源スイッチが OFF になっていることを確認したうえで，電源プラグをコンセントに差し込みます．そして，最初に周辺装置の（ディスク，プリンタ）の電源を入れましょう．次に本体とディスプレイ装置の電源を入れます．しばらくすると，画面にオープニングメッセージが表示されるはずです．(表示されない場合には，PC-8801 本体にある RESET と書かれたスイッチを押してみます．これで，ほとんど表示されるはずです．)表示されない場合には，すみやかに，ディスプレイ装置の電源スイッチを OFF にし，次に，PC-8801 本体の電源スイッチも OFF にして，もう 1 度，各装置の接続の仕方が本当に合っているか調べます．たぶん，間違っているのではありませんか．正しく接続仕直してもう 1 度やり直しましょう．

オープニングメッセージの感じはどうでしょうか．これから先，あなたが PC-8801 の電源を入れる度に表示されるメッセージですから，すぐに顔なじみになるでしょう．そのメッセージの下に，何やらチカチカと点滅している部分がありますね．四角い形をした部分が，ついたり消えたりしています．これを，カーソルと呼びます．これから先，非常によく登場する言葉ですから，覚えてしまいましょう．カーソルです．覚えましたか．

これで，PC-8801 上で $N_{88}$-BASIC が動き始めた訳です．したがって，$N_{88}$-BASIC を通して，あなたは，PC-8801 とやりとりができるようになりました．せっかくですから，何か話しかけてみましょう．あなたの目の前にあるタイプライ




タの様な物から，次のようにキーを１つずつ選んで押してみます．ゆっくりとあわてずに．

HELLO

タイプライタのキーを押すと，そのキーに書かれている文字と同じ文字が画面に表示されましたね．HELLO また，小文字で hello と表示されたと思います．今は，大文字でも小文字でもかまいません．気にしないでいきましょう．今は気にしません．画面に小文字，あるいは大文字の英字が表示されずに，カナカナが表示されてしまった方もいるのではありませんか．その時には，あわてずに，キーボードから，カナと書かれているキーを探して，１度そのキーを押すことで，今度は英字が表示されるはずです．表示されてしまったカタカナは，そのままにして続けて HELLO 】と押してみましょう．小文字の英字が表示されている方の中で，どうしても本書に書かれているのと同じように，大文字にしたいのであれば，キーボードから CAPS と書かれているキーを探して，押してみます．今度は，きちんと大文字で表示されると思います．

】の書かれているキーを押したとたんに，ブザーの音がして，次の行に PC-8801 はメッセージを返してきました．これは，PC-8801 が，「hello ってどういう意味だかわからないよ」と，言い返したきたのです．しかし，気にする必要はありません．それでは，タイプライタに慣れていただくために，図の斜線のついているキーの中から，どれでも構いません．選んで押してみてください．どんな文字が画面に現われるでしょうか（斜線のついているキー以外は，まだ押してはいけませんよ）．

## キーボードを操作する

PC - 8801 とのやりとりは，どうしても，あなたの目の前にあるタイプライタの形をしたものを通して行われることになります．このタイプライタのことを，こ




れからはキーボードと呼びます．覚えてください．ここでちょっと，キーボードの端から端まで，ながめてみてください．

たくさんキーがありますね．全部でいくつあると思いますか．数えてみたら，合計92個のキーが並んでいました．これだけキーがあると，どこにどのキーがあるのか，覚えるのは大変なことです．しかし，ある程度覚えてしまわないと，押したいキーを探すのに時間がかかってしまい，しまいには，面倒だからといって，PC－8801に実行させたいことがあっても，やらせずに済ませてしまうようになるかもしれません．これは，最初にお話しした「やってみよう」精神に反することです．そこで，少しでもあなたにキーボードを操作するのをうまくなってもらいたいので，このキーボードにあるすべてのキーの説明をしていこうと思います．退屈かもしれませんが，じっくりと読んで，キーボード操作を上達させて下さい．

それでは，やっていきましょう．その前に，キーボード上のキーを図のように①，②，③の3つの部分に分けましょう．

①は，キーボードの最上段の横一列のキーの集まりです．合計9個のキーですが，これは，特殊キーと呼ばれるもので，この特殊キーに関しては，後からそれぞれ説明されていますので，簡単な説明にとどめましょう．

### STOP キー

まず，STOPと書かれたキーがあります．何をストップ（停止）させるのかというと，PC－8801を止める時に使います．別にPC－8801が，かけずりまわる訳はないのですが，一般にPC－8801上で走っているプログラムを止めるのに用います．急にプログラム等という言葉が出てきて，よくわからないと思いますが，単にPC－8801を止めるためのキーだと考えておいてください．

### COPY キー

その隣りにはCOPY(コピー)と書かれたキーがあります.このキーは,もしあなた

のPC－8801にプリンタがつながっていないのであれば,押す機会はないでしょう．ですから，やたらに押してはいけません．もし，プリンタにつながっていないのにこのキーを押してしまった時には，すみやかに，先ほどのSTOPキーを押します．そうしないと，それ以降PC－8801は，あなたの言う事を聞いてくれなくなってしまいます．プリンタがつながっているのであれば，プリンタの電源をONにして，このキーを押してみるのが，このキーの働きを知る最良の方法です．

では，何をコピーするのかというと，これは，あなたの見ている画面に表示されているものを，プリンタという装置を使って，紙（リスト用紙）に印字します．ですから，現在見ている画面を，後々まで残しておきたい時には，プリンタをつないだうえでこのキーを押せば，画面の内容が紙上にコピーされます．このコピーのことを，ハードコピーと呼び，後から何度でも見ることができます．逆に，画面上に表示されているものは，ソフトコピーと呼ばれ，こちらは，いつまでも残しておく訳にはいきませんね．あなたが，キーボードのキーを押せば，押されたキーに書かれている文字が，画面に表示されて，それまで表示されていた画面とは違うものとなってしまいます．

### ファンクションキー

さて，COPYキーの右隣りには，ちょっと大きめのキーが5つ並んでいます．このキーのことを，ファンクションキーと呼びます．APPENDIX Aに，このキーの説明はされていますので，詳しい説明は，そちらにゆずります．

このファンクションキーとは，あなたが自由に定義できるキーなのです．つまり，たとえば，このキーを，ABCDと定義しておくと，あなたは，その定義したファンクションキーを1回押すだけで，一遍に4文字分（ABCD）をキーボードから押したのと同じ事をさせることができます．キーボードに慣れていない方にとって，このキーは非常に強い味方となります．また，あなたの定義できるキーの個数は見ての通り，合計10個あります．えっ，5個しかキーが並んでいないって，確かにキーの個数は5個しかありませんが，心配入りません．ちゃんと10個分定義できるのです．その秘密は，もう少し後から出てきます．お楽しみに．

### ROLL UP，ROLL DOWN キー

ファンクションキーの隣りには，ROLL UP,ROLL DOWNと書かれたキーが2つ並んでいますね．どうぞ押してみてください．どちらでも構いません．ただし，画面に何も文字が表示されていないと，このキーを押しても，その働きはわからないでしょう．その時には，キーボードから何か文字を数行入力した後で，

押してみます．非常に，面白いキーだということがわかっていただけましたか．詳しい説明は，ユーザース・マニュアルを見てください．

②は，見ればわかるように，非常にたくさんのキーがあります．そこで，まずキーの色の違いに注目してください．こげ茶色のキーと白色のキーに分かれていますね．自分のキーボードで確かめてみましょう．このこげ茶色のキーは，特殊キーと呼ばれているもので，後から詳しい説明がされているキーもありますので，ここでは，そのキーの名前と，簡単な働きの説明に止めます．白色のキーに関しては，こげ茶色のキーを説明していく中で，実際に押してもらいながら説明していきます．

### ESC キー

では，左側の上にある ESC と書かれたキーの説明に入りましょう．このキーは，エスケープキーと呼ばれています．このキーは後で説明する CTRL キーと同じように他のキーと同時におすことによってキーの数が多くなるような効果を与えます．しかし BASIC モードではほとんど使うことがありませんから，気にせず次に進みましょう．

### TAB キー

ESC キーの下には，TAB (タブ) と書かれているキーがありますね．タブキーと呼びます．論より証拠．押してみます．カーソルの位置が動きましたね．もう１度押してみます．またカーソルが動きました．では，今度は，白色のキーの中から，１番大きなキーを探してください．そうです．キーボードの１番下にある横に長いキーのことです．このキーは，スペースキーと呼ばれます．このキーも，１度押してみます．

カーソルが，横に１つ分動きましたか？10秒ほど，このスペースキーを押してみましょう．カーソルが連続して動きましたか？TAB キーでも，スペースキーでも，このようにカーソルを動かすことができますが，TAB キーの方が大きく動かすことができますね．でも，スペースキーを押し続けた時には，カーソルがその行の右端に来ると，今度は１段下の行の左端に戻りましたね．同じ行を動いた訳ではありませんよ．１段下の行に移りました．しかし，TAB キーを何回か押して，カーソルが右端に来てしまうと，それ以上は，TAB キーをいくら押してもカーソルは動かなくなります．

実際にやってみましょう．TAB キーを，最低５回押してみます．カーソルは右端に来ましたね．さらにもう１回 TAB キーを押します．でもカーソルは動かない

でしょう．では，スペースキーを１回押します．カーソルは，次の行の先頭で点滅しているでしょう．

次に，キーボードの右側の数字の書かれたキーの集まりの中から１と書かれたキーを押します．そして，TAB キーを１回押して，今度は，２と書かれたキーを押します．そして，TAB キーを１回押します．さらに，３を押して，TAB キー，４を押して，TAB キー，５を押して，TAB キー，６を押すと，カーソルは次の行の先頭に戻りますね．１つ上の行には，今押していった１から６までの数字が表示されていますね．この数字の位置と，カーソルの位置に目をやりながら，TAB キーを１回押してみます．何と，カーソルは，２の表示されている下の行に来ましたね．もう１度，TAB キーを押してみます．今度は，３の下に来ました．このように，TAB キーを使うと，いつでも，２から６の数字の表示されている位置まで動かすことができるのです．つまり，簡単に桁合せができるのです．今はまだピンときませんが，あなたがプログラムを入力するようになると，きっと使うようになるキーですから，覚えておきましょう．ちなみに，前に登場したファンクションキーの説明がわかった方には，この TAB キーには，スペースキーを８回押したのと同じ働きを定義してあるという言い方をした方が，わかってもらえるかもしれません．

### CTRL キー

TAB (タブ) キーの下に，CTRL と書かれたキーと，CAPS と書かれたキーがありますね．まずは，CTRL と書かれたキーの説明をしていきましょう．このキーは，コントロールキーと呼ばれています．キーボードには，合計92個のキーがあることは前に書きました．しかし，この92個だけではまだ足りないので，もっとキーの数を増やしたいのですが，それこそ，キーを探すのに手間がかかってしまいます．そこで，このキーを押した状態の時と，押していない状態の時の２つに分けて，キーボードにあるキーを２倍にしようとするのが，このキーの働きです．つまり，CTRL キーを押した状態で２つ隣りにある A と書かれたキーを押すのと，CTRL キーを押していない状態で A と書かれたキーを押すのとでは，違う働きをさせようというものです．先程は，キーボードのキーが２倍になるなどと書きましたが，実際には，この CTRL キーによって合計18個キーが増えるだけです．逆に言えば，キーボードにある18個の特別なキーだけに関しては，この CTRL キーと併用することで，全く別の働きをさせることができる訳です．

18個の特別なキーに関しては，後から，特殊キーの所で，説明していきます．しかし，一つぐらいやってみましょう．まずキーボードから，G と書かれたキー

を探して押してみます．画面に，Gあるいはｇと表示されるだけですね．では，今度は，CTRLキーを押したまま，Gと書かれたキーを押してみます．どうです．PC－8801に内蔵されたブザーが聞こえてきましたか．何度でも，あなたの気の済むまでやってください．残り17個のキーに関しては，もう少し後までお楽しみに．

### CAPSキー

CTRLキーの隣りには，CAPSと書かれたキーがありますね．このキーはキャピタルロックキーと呼ばれています．現在，画面に小文字の英字が表示されているのであれば，このCAPSキーを1回押すことで，今度は大文字の英字が表示されるようになります．やってみましょう．今まで，大文字の英字が表示されていたのであれば，小文字に変わりましたね．もう1度CAPSキーを押せば，元に戻ります．これからの説明の都合上，大文字の英字が表示されるように，CAPSキーをセットしておきましょう．CAPSキーを押してみるとわかりますが，他のキーと押した感じが違っていて，いかにもスイッチという感じがしませんか．カチッ，カチッと，なるでしょう．このようなキーは，他にもう1つあるだけです．カナと書かれたキーがそうです．

### SHIFTキー

CTRLキーと，CAPSキーの下には，少し大きめのキーがあります．シフトキーと呼ばれています．気が付いたでしょうか．そのSHIFTキーの右側を見ていくと，右側にも，同じSHIFTと書かれたキーがあります．この2つのキーは，どちらも全く同じです．左側にあるか右側にあるかの違いだけですから，気にせずに，どちらかあなたの押しやすい方を押せばよいでしょう．

では，押してみます．画面をよく見ていてください．すぐに放してしまわずに，押したり放したり、何度かやってみます．画面の1番下に白黒が反転している行がありますね．その中に表示されている文字が，このSHIFTキーを押すと，変わるでしょう．実はここに表示されているのが，前に登場したファンクションキーに定義されている内容だったのです．そして，ファンクションキーは5個しかないいのに，合計10個あると不思議な事を書きましたが，このSHIFTキーを使うと納得できます．つまり，SHIFTキーを押さないでいる状態では，ファンクションキーは，そのキーに書かれているように，f・1からf・5まで，5個ありますね．そして，SHIFTキーを押した状態にすると，今度は，f・1と書かれているキーは，f・6の意味になります．したがって，f・2と書かれているキーは，f・7となり，f・5がf・10になって，めでたく合計10個のファンクションキーが，お目見えす

る訳です．わかりましたか．ファンクションキーには，f・1からf・5までしか書かれていませんんが，SHIFTキーを押した状態では，このf・1の1を6と見ます．f・2の2は7と見ます．大丈夫ですね．

しかし，SHIFTキーは，ファンクションキーを合計10個にするだけの働きしかないのかというと，そうではありません．同じことが，②の部分にある白色のキーに対しても適用されます．この白色のキーには，英字の他に，カナ文字も書かれていますね．また，特殊文字と呼ばれるものが，1つのキーの中に書いてあるものもあります．1つだけキーを例に上げて説明しますので，その他の白色のキーに関しては，あなたが調べてみてください．

ESCキーの3つ隣りに，3と書かれているキーがありますね．もちろん，3の他に，#，ア，ァと，合計4つの文字が書かれています．そこで，キーボードを説明するうえで，全体を3つに分けたように，このキーに書かれている部分も，4つに分けます．他の②の部分の白色のキーについても同じです．1つのキーを，図のように三角形が4つ集まって出来ていると考えます．




そして通常，キーを押した時には，3と書かれている部分の文字が表示されます．やってみましょう．3が画面に表示されますね．次に，SHIFTキーを押した状態で，押してみます．今度は，#が表示されました．他のキーについても同じことですから，同様に考えますが，この4つの部分に必ず何か文字が書かれているとは限りません．

たとえば，7つ左側に0と書かれたキーがあります．カタカナの，ワとヲも書かれていますが，先程の3と書かれているキーの#が書かれている場所には，何も書かれていませんね．試しに，SHIFTキーを押した状態で，この0のキーを押してみます．カーソルが1つ右へ進んで，スペースが表示されているようですが，あてになりませんので，これは使わない方がよいでしょう．スペースを入力したければ，必ずスペースキーから入力しましょう．ここで，キャピタルロックをはずしてください．大文字の英字の書かれているキーを押すと，小文字の英字が表示されます．しかし，キーには，小文字の英字は書かれていません．これは，わざわざキーに書かなくてもわかりますね．また，大文字と小文字の両方を書くと，ごちゃごちゃして見にくくなるためでしょう．SHIFTキーを押した状態で，英字の

書かれているキーを押すと，何が表示されるでしょうか．やってみます．SHIFT キーを押した状態で，CAPS キーの隣りにあるＡと書かれたキーを押してみます．何と，大文字のＡが表示されました．SHIFT キーを放して，Ａのキーを押してみると，当然小文字のａが表示されます．何か気が付きませんか．これで，CAPS キーの働きがわかったのではありませんか．CAPS キーは，この SHIF キーを押した状態にしておくか，放した状態にしておくか，選ぶことができるのです．ですから，英字を大文字でいくつも入力したい時には，SHIFT キーを押しながらやるのではなくて，CAPS キーで，SHIFT キーを，押した状態にしておき，入力すればよいことになります．では，CAPS キーを押して，英字のキーを押した時には，画面に大文字の英字が表示されるようにします．この時，SHIFT キーを押した状態で，何かキーを押すと，表示されるのは，一体何でしょうか．試しに，CAPS キーの隣りのＡのキーを，SHIFT キーを押した状態で押してみます．何と，小文字のａが表示されたでしょう．わかりましたか．SHIFT キーを押した状態に，CAPS キーをしておき，実際に SHIFT キーを押すと，今度は，SHIFT キーを押しているのにもかかわらず，SHIFT キーを放した状態になるのです．何となく複雑ですから，たぶん読んだだけではわからないでしょう．実際にキーボードを押して，確かめましょう．理解できたら，次へ進みましょう．

### カナキー

SHIFT キーの下の右にカナと書かれたキーがありますね．カナキーと呼びます．前に書いたように，CAPS キーと同じで，スイッチという感じのするキーです．このカナキーに関しては，これまで書いてきたことが，充分わかっていただいていれば，難しいことではありません．もう１度，キーを４つの部分に分けた図を書きます．

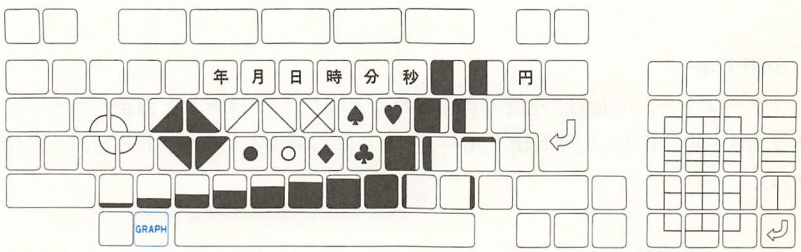



図の青色の部分が，今まで SHIFT キーを押した時に対象となっていた部分ですが，今度はこのカナキーを押すことで，青色以外の部分に書かれている文字が対象となります．SHIFT キーを押した時には，どちらの三角形の部分に書かれている文字が表示されるのかは，あなたにお任せします．やってみれば，わかりますね．表字される文字が，英字からカナ文字に変わるだけで， SHIFT キーの使い

方も先程と同じです．もうわかっていますね．

### GRAPH キー

カナキーの隣りには，GRAPH と書かれたキーがありますね．グラフィックキーと呼ばれています．このキーも，SHIFT キーや CTRL キー等と同じように，このキーを押した状態にして，他のキーを押す必要があります．試しに，GRAPH キーを押した状態で，カナキーの上にある Z と書かれているキーから順番に右側のキーを１回ずつ押していってみます．何と，グラフィックシンボルと呼ばれる，今まで見たことのない文字が表示されましたね．このグラフィックシンボルは，キーボードのどこを見ても書かれていません．そこで，どのキーを押すと，どんなグラフィックシンボルが表示されるのかを示すために，キーボード図を載せておきますので，GRAPH キーを押した状態で，いくつか押してみて，画面に表示させてみましょう．ただし，」と書かれたキーが２つありますが，このキーを押しても，画面に，」の記号が表示されるのではありません．念のために．また，何もグラフィックシンボルが書かれていないキーに関しては，押してみても何も表示されません．




### INS・DEL キー

さて，②の部分に含まれる特殊キーも，残り７個になりました．今後は，②の部分の右側を見ていきましょう．まず最初に，上段に INS，下段に DEL と書かれたキーがありますね．インサート・デリートキーと呼ばれています．このキーの説明は，後から詳しく出てきますので，省略します．１つだけ注意する点は，これまで１つのキーを４つの部分に分けて説明しましたが，このキーは上段と下段の２つの部分に分かれていますね．これは，カナキーを押している時と，押していない時のどちらでも，同じ作用をすることになります．通常，SHIFT キーを押

せばINSキーの働きをし，SHIFTキーを押さなければDELキーの働きをすることになります．

### リターンキー（ = ）

と書かれたキーが，INS・DELキーの下にあります．リターンキーと呼ばれています．キーボードを気を付けて見ると，③の部分にも，右端に，大きさは違いますが，と書かれているキーがあります．どちらも，全く同じ働きをしますので，あなたの押しやすい方を押してください．大きな方が，押しやすいでしょうか．働きに関しては，ここでは省略します．というのは，もう少し，PC－8801に慣れていただいた後から，説明した方が説明しやすいためです．




### SHIFTキー

リターンキーの下には，左側で登場したのと同じく，SHIFTと書かれたキーがあります．前に登場したSHIFTキーと，全く同じ働きをします．もうおわかりですね．

### カーソル移動キー

SHIFTキーの下には，何やら四方向の矢印の書かれたキーが，SHIFTキーを取り巻く様にありますね．このキーの働きの説明は，後にゆずるとして，1つ気が付くことは，上向きと，右向きの矢印の書かれているキーには，おまけとして，その矢印の下に小さな黒丸がついています．これは何を意味しているのかというと，$N_{88}$－BASICが走っている場合には，全く関係がないのですが，N－BASICが走っている時には，少なからず関係します．というのは，PC-8001をお持ちの方ならばおわかりのように，PC-8001では，この4つのキーは，2つにまとめられています．つまり，左右の矢印と上下の矢印のキーがそれぞれ，1つのキーに書かれています．そして，SHIFTキーを押した状態と，そうでない状態とで，2つのキ

ーを4通りの働きに使っています．PC-8801では，N-BASICを走らせている場合には，$N_{88}$-BASICと同じように4 つのキーを使うことも，2 つのキーと SHIFT キーを組み合わせて使うことができます。その対応は，下の通りです．N-BASICを走らせる場合には，気を付けてください．

| | | |
|---|---|---|
| →のキーの働きをさせるには？ | ⟶ SHIFT キーを押さずに， | [→] と書かれているキーを押します． |
| ←のキーの働きをさせるには？ | ⟶ SHIFT キーを押しながら， | [→] と書かれているキーを押します． |
| ↑のキーの働きをさせるには？ | ⟶ SHIFT キーを押さずに， | [↑] と書かれているキーを押します． |
| ↓のキーの働きをさせるには？ | ⟶ SHIFT キーを押しながら， | [↑] と書かれているキーを押します． |

さて，これで，②の部分にあるすべてのキーの説明は終わりました．残るは，③の部分の説明だけです．その前に，1つだけ書いておきたいことがあります．それは，今まで，経験された方もいるかもしれませんが，TAB キーの説明の所で，スペースキーを押した時のように，1つのキー（CTRL キーと SHIFT キーは除きます）を，少し押し続けていると，そのキーが連続して入力されていきます．ですから，同じキーを連続して何度も入力しなければならない時には，ポン，ポンと，いちいちキーを1度放して入力していく必要はなく，そのキーを押し続ければ，好きなだけ入力できるという訳です．

③の部分の説明に入りましょう．いよいよ，キーの説明の最後です．がんばりましょう．この③の部分のキーは，俗にテンキーと呼ばれるキーの集まりです．合計20個のキーがありますが，テンキーと呼びます．こげ茶色と白色の2色に分かれていますが，このテンキーの中で，特殊キーと呼ばれているのは，3つだけです．わかりますね．左側上から，HOME・CLR と書かれたキーと，その隣りの HELP と書かれたキー，そして最後が，前にも登場した ↵ と書かれているリターンキーです．その他のキーに関しては，1つにまとめて説明していくことにします．

### HOME・CLR キー

ホーム・クリアキーとでも呼びましょう．テンキーの中で，唯一 SHIFT キーの影響を受けるキーです．すなわち， SHIFT キーを押した時には，HOME キーの

働きを，SHIFT キーを押していない時には，CLR キーの働きをする事になります．

HOME キーの働きですが，画面上には，ホームポジションと呼ばれる位置があります．どこかというと，画面の左上の隅です．そこで，SHIFT キーを押しながら，このホーム・クリアキーを押すと，瞬時に，カーソルが，画面の左上隅（ホームポジション）に来ます．実際にやってみてください．1度，ホームポジションにカーソルが来てしまえば，後は何回やっても同じですから，TAB キーやリターンキーを使って，一旦カーソルを移動させた後で，再びやってみましょう．SHIFT キーを押しながらホーム・クリアキーを押すのを，忘れないように．すでに，SHIFT キーを押さずに，このキーを押してしまった人は，CLR キーの働きがわかったはずです．まだの人は，SHIFT キーを押さずに，ホーム・クリアキーだけを押してみます．それまで，画面に文字が表示されていたのであれば，瞬時に画面に表示されていた文字が消されて，カーソルがホームポジションに来たはずです．画面に何も表示されていなければ，適当にキーを押して，何かしら画面に文字を表示させた後で，ホーム・クリアキーを押します．どうです．画面に表示されていた文字がきれいに消されたでしょう．したがって，SHIFT キーを押した状態でこのホーム・クリアキーを押すと，カーソルがホームポジションと呼ばれる画面の左上隅に来ます．SHIFT キーを押さずにホーム・クリアキーを押すと，画面に表示されていた文字がすべて消された上で，カーソルがホームポジションに来る訳です．おわかりでしょうか．

### HELP キー

ホームクリアキーの隣りに，HELP と書かれているキーがありますね．ヘルプキーと呼びます．何を助けてくれるのかというと，今の段階では，説明しても，あなたがプログラムというものをあまりよく知らないのであれば，無駄になってしまうかもしれませんので，簡単な説明だけにしておきます．ですから，あなたが，本書の 3 章以降を読まれたうえで，リファレンス・マニュアル等で，この HELP キーの説明を読まれるとよいと思います．

このキーの使い方は，いくつかありますので，その中の 1 つだけ説明しましょう．あなたがプログラムを作っていて，そのプログラムを実行させた時に，もし何かしらエラーメッセージを，PC - 8801 が返してきたとすると，この HELP キーを押すと，プログラムのどこでエラーが起きたのかを，カーソルの位置により教えてくれます．必ずしも，そのカーソルのある場所に間違いがあるとは限りませんが，エラーの起きた場所を知るうえで，大変役に立つキーだと思います．

### テンキー

さて，残りのキーは，リターンキーを除いてまとめて説明しましょう．と，いっても，それぞれのキーに書かれているキーが，画面に表示されるだけです．特別な働きがある訳ではありません．②の部分にも，数字を入力できるキーがあるのに，何故ここにもあるのか不思議に思われるかもしれませんが，あなたが，数字をたくさん押すことがあった時には，こちらのキーを押した方が押しやすいですし，また能率良くできるのではないでしょうか．また5と書かれたキーを，ちょっと触ってみてください．キーの表面に，ボッチが付いています．このボッチにより，5のキーだということがキーボードをいちいち見なくてもわかるようになっています．5のキーの回りに，1から9までの数字がありますので，常に5のキーを基準に考えればよいでしょう．ただ，0のキーは少し離れた所にありますので，気を付けましょう．，（コンマ）と．（ピリオド）も②の部分にありますが，こちらの方がすぐに見つかりますね．多いに利用してください．－，／，＊，＋，＝についても同様です．

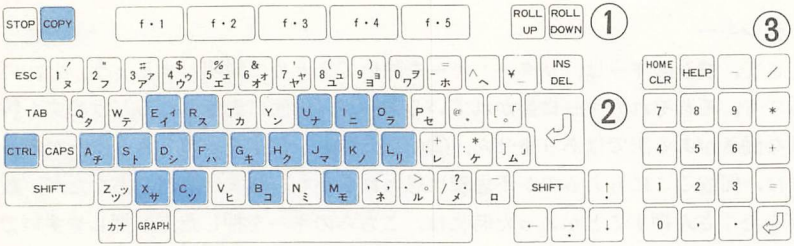
### リターンキー

②の部分にあったキーと，全く同じ働きをします．小さい分だけ，少し押しにくいかもしれませんが．同じ様にリターンキーと呼ばれています．

以上，キーボードのすべてのキーについて，説明してきました．これから，あなたがお世話になるキーボードです．余り強くキーを叩いたりすると，PC-8801のためにも，あなたの指のためにもよくありませんから，キーはていねいに押していきましょう．

## 特殊キーにはこんなものが

では，これから特殊キーの説明をしていきます．しかし，すべての特殊キーの説明をする訳ではありません．CTRLキーの所で出てきた18個のキーについての説明を，これからしていきます．まずは，次ページのキーボードの図を見てください．色が違うキーが，CTRLキーを除いて合計18個ありますね．もちろん，前にやったことのあるGのキーも，その中に含まれています．この色の違うキーに限って，CTRLキーと合せて使うと，別の働きをさせることができます．と，書きましたが，実際には，この18個のうちの7個は，すでにキーボードにあるキーと同じ働きをするのです．したがって，残りの11個だけが，今まで登場していない働きを持つことになります．では，先に，すでにキーボードにあるキーと同じ




働きをする7個のキーについて見ていきましょう．

CTRL-A ：HELP キーに同じ
CTRL-C ：STOP キーに同じ
CTRL-H ：DEL キーに同じ
CTRL-I ：TAB キーに同じ
CTRL-M ：↵（リターンキー）に同じ
CTRL-K ：HOME キーに同じ
CTRL-L ：CLR キーに同じ

以上，7個がそれぞれ後ろに書かれたキーと同じ働きをします．ですから，この7個に関しては，わざわざ覚えなくてもすでに覚えているキーを使えばよいのです(わかっていると思いますが，たえとえば，CTRL-A とあるのは，CTRL キーを押した状態で，英字の A の書かれたキーを押すという意味です)．

次に，残りの11個に関しては，それぞれ1つずつ説明していきましょう．

### CTRL　COPY

これは，CTRL キーを押した状態で，COPY とキーを押す意味ですね．何が出来るのかというと，COPY キーの所の説明では，画面に表示されているものをプリンタを使って，紙（リスト用紙）にコピーできると書きましたが，CTRL キーと一緒に押すと，画面に表示されている文字をコピーできます．これではよくわかりませんね．無理もありません．あなたは，まだ画面に文字以外のものを表示したことはありませんし，見たこともないでしょう．

実は，PC-8801 は，今まであなたが見ていた文字の他に，画面に細かな点をたくさん打つ事ができるのです．つまり，画面に打った点の集まりで，絵を書く事

ができます．よくわからない方は，次のように考えてください．今，あなたの見ている画面は，実は2枚の同じ大きさの画面を重ねて見ているのです．その中の1枚には，文字が表示されています．あなたが，キーボードを押した時には，その押された文字が画面に表示されますが，この文字は，2枚のうちの1枚の画面に表示される訳です．では，残りのもう1枚の画面はというと，現在何も書かれていません．したがって，二枚の画面を見ているのですが，実際には1枚の画面だけを見ているのと同じことです．そこで，現在何も書かれていない方の画面にも何か書いてみたくなりますが，これは，本書の第4章でやっていますので，それまでお待ちください．

こちらの画面には，たくさんの点を打つことができると前に書きましたが，その点の個数だけ，書いてみると，横方向には最高640個の点を打てます．縦方向には，最高400個または，その半分の200個です．したがって，画面全体では最高640×400＝256000個．または640×200＝128000個の細かい点を打ったり消したりできる訳です．ちょっと，気が遠くなりそうな数ですね．詳しい事は，第4章をお読みください．

さて，話を元に戻して，COPY キーだけを押した時には，この2枚の画面を重ねて紙（リスト用紙）にコピーすることができます．ですから，あなたの見ている画面と全く同じ画面をコピーできます．CTRL キーを押しながら COPY を押すと，今度は文字が表示されている方の画面だけがコピーできます．したがって，細かい点を打つ事のできる画面の方は，仮にあなたの見ている画面に表示されていたとしても，コピーされないという訳です．

CTRL-B
CTRL-D
CTRL-E
CTRL-F
CTRL-J
CTRL-O
CTRL-U
CTRL-X

これらの機能は，APPENDIX A の所で説明されています．そちらを見てください．

### CTRL-G

これは，以前に出てきました．もう，御存知ですね．PC-8801 に内蔵されたブザーを一定時間鳴らすことができます．

### CTRL-S

これは，表示される画面を一時停止させることができます．つまり，あなたの見ている画面は，ソフトコピーと呼ばれていると，以前に書きましたが，画面一杯に表示され，さらに表示が続く時には，１番上にある行から順番に消されていき，画面が下から上へと，流れていきます．この流れていくのを止めるには，STOP キーを使えば止まります．しかし STOP（ストップ）キーを押した場合には，実行自体が止まってしまい，続けて実行するには，CONT（コンティニュー）という特別な命令を使わなければなりません．CONT などという知らない命令が出てきましたが，慌てる必要はありません．後から詳しく説明されていますので，その時に覚えましょう．CTRL-S は，画面の表示を一時止めておくことができるのです．さらに，もう１度，CTRL キーを押しながら，S と書かれたキーを押せば，表示が再開されます．まだ，画面の表示が下から上へ流れていくというのが，よくわからないかもしれませんが，画面の表示を一時止めておくには，CTRL -S を押すと覚えておいてください．

以上，合計18個の説明が終わりました．その内の８個については，APPENDIX A を見てください．というのは，今の段階でこの８個のキーの説明をしても，たぶん何の事かわかってもらえないと思うからです．したがって，もう少し後から，そうですね，あなたがプログラムというものをある程度理解した時に，改めて，読むとよいでしょう．ちなみにこの８個のキーについては，知っていると便利なものですが，別に知らなくてもどうということはありません．

## 目的のキーを探す

以上が，あなたの目の前に置いてあるキーボードの説明です．キーボード操作をするうえで，少しは役に立ったでしょうか．とにかく，キーの種類がたくさんあって，お手上げですか？もうそんなことはありませんね．たくさんあるキーも，すでにあなたの目には，特殊キーと，そうでないキーとに分かれて見えるでしょう．使い方も，ある程度わかったことと思います．後は少しでも速く，押したいキーを見つけられるようになれば，PC - 8801 とのやりとりも，いっそう楽しいものになることでしょう．別にタイピストの様にキーボードが打てるようになる必

要はありませんが，押したいキーを探すうえでのアドバイスを，少ししておきましょう．

1)どうしてキーボードには英字のＡからＺまで規則正しく並んでいてくれないのでしょう．などと，あなたは考えていませんか．その通りです．ＡからＺまで，また，カナ文字のアからンまで，規則正しく並んでいてくれたらどんなにいいでしょう．しかし，実際のキーボードには，ばらばらに並んでいるように見えます．でも，これは，タイピスト達にとって，左右の指を公平に使えるように並んでいるらしいのです．普通タイピストは，親指を除いた８本の指を常にキーボードの上においています．知っている方もいるでしょう．左手の小指は，Ａの位置に，人指し指はＦに，Ａ・Ｓ・Ｄ・Ｆの上に指が乗せられています．右手の指は，人指し指が，Ｊの上に，小指は；の位置に，Ｊ・Ｋ・Ｌ・；の上に乗せられています．あなたも，ちょっとやってみましょう．常にこの状態から，指をのばして，いろいろなキーを押しているのです．何となく，ＡからＺまでのキーの配置が，どうしてそうなっているのかわかるような気がしませんか．

2)数字を打つ時には，迷わず，テンキーにある数字を打ちましょう．あなたがタイピストでないならば．

3)カーソル移動キーの４つの矢印の書かれているキーは，どんなことをしても，位置を覚えましょう．つまり，画面から目を放すことなく，どの方向のキーでも，押せるようになるのがよいでしょう．これだけでも，能率が上がりますよ．

4)ファンクションキーは，大いに活用しましょう．あなた自身が，ファンクションキーの定義をして使うのもよいでしょう．PC－8801の電源を入れた状態の時には，10個のファンクションキーは，それぞれ使いやすいように定義されています．キーボードからキーを探すのに時間のかかる人は，自分の押したい文字がファンクションキーに定義されていないか，まず調べるのもよいかもしれません．

5)英字のＡからＺまでが，ある程度すぐに見つかるようになったら，次には，カナ文字にも挑戦しましょう．ちょっと数が多くなりますが，数が多い分だけ，探すのに時間がかかる訳です．特に，プログラムの中でカナ文字を多く使う人は，やっておくとよいでしょう．

6)英字の小文字と大文字については，それほど気にすることはありません．極端な話をすると，プログラムであなたが入力した小文字の英字は，特別な場合を除いて，PC－8801は，すべて大文字に変換してくれます．ですから，これから先，あなたにキーボードから入力してもらうことが度々ありますが，その時には，次の場合を除いて，英字に関しては，大文字でも小文字でも構いません．その場合とは，その英字が〝(ダブルクォーテーションマーク)で囲まれる中にある場合のことです．

7)ROLL UP, ROLL DOWN キーを有効に使いましょう．特にこれから先，画面にグラフィックスを表示させるようになると，2枚の画面が重なって，文字が見えにくくなることがあります．その時には，これらのキーで，上下に移動させてやると多少見易くなるでしょう．

8)HELP キーもその使い方を覚えて，大いに利用しましょう．

## コンピュータとカセットレコーダ

PC－8801には，御存知の様にカセットテープレコーダをつなげることができます．もちろん，フロッピーディスク装置をお持ちならば，カセットテープレコーダの替わりをさせることができます．また，カセットテープレコーダよりも正確に，早く同じ事ができます．別に，カセットテープレコーダを使うと，正確にできない訳ではないのですが，それには，いくつか注意しなければならないことがありますので，これから，それらのカセットテープレコーダを使ううえでの注意を書いていきます．カセットテープレコーダをお使いになる方は，しっかり読んでおきましょう．

すでに，PC－8801とカセットテープレコーダは，接続ケーブルによって，つながっているでしょうか．まだだという方は，今のうちに付属の取扱い説明書等で，つなぎ方を調べたうえで正確に接続しておきましょう．何事も，初めが肝心です．間違いないように．

用意したカセットテープレコーダは，きちんと録音と再生ができるでしょうか．ヘッドが汚れていたりしたら，今のうちに掃除しておきましょう．

また，テープデッキをお使いになる方は，左右どちらかの片方のチャンネルのみ使われるのがよいと思います．何度か自分で試してみて，最良の方法を探しましょう．できれば，モノラルのカセットテープレコーダを使うのがよいでしょう．ないのであれば，仕方ありませんが……．

ここで，少し難しい話しになりますが，カセットテープレコーダのテープスピードは変える事はできませんが，PC－8801からカセットテープレコーダに送る信号の変調速度に関しては，PC－8801は2種類の変調速度を選ぶ事ができます．この変調速度は，ボーという単位で表わされ，数が大きくなるほど，変調速度が大きくなります．PC－8801では，600ボーと1200ボーの2種類の変調速度を使う事ができます．どうして2種類も持っているのかというと，PC－8001をお持ちの方ならばすぐにわかりますね．PC－8001の変調速度は，600ボーでした．したがって，PC－8801を使ってPC－8001で録音したカセットテープを読む場合には，600ボーで読まなければなりません．ですから，2種類の変調速度を持っている訳です．しかし，2種類あるということは，カセットテープを読む時に，どちらの変調速度を使うかという指定をしなければなりません．この指定には，カセットテープを読むのであれば，

```
LOAD "CAS1:------"
```

と，

```
LOAD "CAS2:------"
```

というように区別します．CAS1：を指定すると，変調速度は1200ボーが選ばれます．CAS2:を指定すれば，600ボーという事になります．$N_{88}$-BASICを使っている場合には，CAS1:，つまり，1200ボーで使う事が多くなります．付属のカセットテープも，1200ボーで録音されています．ですから，カセットテープを読む場合には，

```
LOAD "CAS1:------"
```

という様になります．間違えないように，気を付けてください．

## サンプルプログラムを読む

それでは，付属のカセットテープを使って，実際にカセットテープに録音されている内容を読んでみましょう．まずは，音量（ボリューム）と音質（トーン）を，どこらへんに合わせればよいのかわかりませんが，音量は，中位に，音質は高目になるように合わせておきます．そして，キーボードのホーム・クリアキーを押して，画面に表示されている文字を消した上で，次のように入力していきます．

```
LOAD "CAS1:DEMO"
```

間違えずに入力できましたか？途中で間違えてしまったならば，もう1度，ホームクリアキーを押す所からやり直しましょう．さて，それでは，カセットテープレコーダにPC－8801に付属のカセットテープをセットします．そして，再生ボタンを押すと同時に，キーボードの ↵（リターンキー）を押します．しばらくして，画面が次の様になれば，

```
LOAD "CAS1:"DEMO"
Found:DEMO
```

うまくいきました．そうでなく，エラーメッセージが返って来たり，全然変化がない時には，カセットテープを，もう1度最初の位置に巻き戻します．そして，エラーメッセージの返って来た人は，音量（ボリューム）を少し上げてみます．そして，先程と同じ様に，キーボードから，

```
LOAD "CAS1:DEMO"
```

と，入力して，カセットテープレコーダの再生ボタンを押すと同時に，↵（リターンキー）を押します．全然変化のなかった人も，音量（ボリューム）を少し上げてから再生ボタンを押します．今度は，どうでしょうか．しばらく待ってみます．また，エラーメッセージが出てしまったり，全然変化がないのでしたら，同じように，音量（ボリューム）を少し上げてから，先程と同じ事を繰り返します．また，最初はうまく読んでいたのに，途中でエラーメッセージの出てしまった人は，カセットを巻き戻してから，同じように音量（ボリューム）を少し上げてもう一度トライしてみます．どこまで音量を上げていっても，エラーメッセージが出てしまうのなら，今度は，音量を少しずつ下げていってみます．それでもだめなようならば，音質（トーン）についても少しずつ変化させてみます．最適な音量（ボリューム）と音質（トーン）が見つかるまで，少し面倒ですが，1度見つけてしまえば，以後は，同じ場所にセットすればたいていうまくいくはずです．

さて，中にはエラーメッセージも出ず，PC－8801からうんともすんとも言ってこない方もいるのではないですか．その方は，本当に正しくPC－8801とカセットテープレコーダがつながっていますか．もう1度，確かめてください．正しくつながっているのであれば，先程の続きで，音量（ボリューム）と音質（トーン）の調整を念入りに繰り返しましょう．

さて，うまく読んでいるようであれば，数分でPC－8801はOkを返してくるでしょう．まずは，カセットテープレコーダの停止ボタンを押して，テープを止めておきましょう．ちなみに1つ書き忘れましたが，リモート端子に，PC－8801からの接続コードをつないでいる方は，思うようにテープを巻き戻せなかったでし

ょう．このリモート端子には，PC－8801からの信号が入っているので，リモート端子がつながっているのであれば，ON・OFFの制御ができます．

```
MOTOR 1 J
```

と入力すると，自由にテープを動かせるようになります．つまり，リモート端子にONの信号が入ります．また，

```
MOTOR 0 J
```

と入力すると，その逆で，テープは止まった状態になります．つまり，リモート端子にOFFの信号が入る訳です．もう1つ，

```
MOTOR J
```

とだけ入力すると，現在の状態がONであればOFFに，OFFであればONとなります．必ずしもリモート端子には，接続ケーブルをつないでおく必要はありません．特に今回のように，テープを巻き戻す必要がある場合は，リモート端子からケーブルを抜いておいた方がよいでしょう．付属のカセットテープをPC－8801がうまく読めるようになるまで，何回でも繰り返し音量（ボリューム）・音質（トーン）の調整をしておきましょう．出来るまで，次へ進んではいけません．根気よく続けましょう．

## サンプルプログラムを実行する

それでは，付属のカセットテープの内容をPC－8801に読ませることができたならば，

```
RUN J
```

と入力するか，あるいは，ファンクションキーのf・5と書かれているキーを押します．これで，今カセットテープから読んできたプログラムの実行を始めます．どうして，カセットテープに，プログラムが入っていたのかなどということは，今の所深く考える必要はありません．もし，あなたの持っているディスプレイ装置がカラーであるならば，当分画面に表示されるきれいな色をお楽しみください．不幸にも，何らかのエラーメッセージが返ってきてしまった方は，カセットテープに録音されていた内容が，正しくPC－8801に入っていませんので，もう1度，先程の所へ戻ってやり直さなければなりません．その時に，音量（ボリューム）を，少し上げてからやり直してください．実は，どうしてエラーメッセージが出

たのかというと，音量（ボリューム）が小さすぎたために，PC－8801がテープに録音されている内容を読み誤まったのです．ですから，もう1度テープを読み直す必要がある訳です．その時に，もし，画面に表示される文字が読みずらかったりしたら，

↵（エラーメッセージが返ってきても，続けて，）

```
CLS 3 ↵
```

と入力すれば，画面に表示されているすべてのものが消え，少し時間をおいた後に，画面の左上隅にOkと表示されて，その次の行にカーソルが点滅されるようになると思います．画面が見づらい時には，やってみましょう．

PC－8801のデモンストレーションは，いかがですか？前にあなたの見ている画面は，2枚の同じ大きさの画面を重ねて見ていると書きましたが，今，あなたの見ている画面は，まさに，その2枚を重ねて見ている訳です．どうです，きれいでしょう．当分は，このデモンストレーションを，お楽しみください．目に焼きついてきた所で，次へ進んで行きましょう．

## サンプルプログラムを止める

デモンストレーションは，目に焼きつきましたか．では，そろそろ，このプログラムを止めましょう．PC－8801上で走っているプログラムを止めるには，どのキーでしたか．そうです．STOPキー，もうひとつは，CTRL-Sでした．

CTRL-Sでは，一時停止させるだけですから，ここではSTOPキーを押して，プログラムの実行をSTOP（停止）させてみます．押してみてください．止まりましたね．画面に表示されているものを消している間，多少時間がかかりますが，どうやら止まったようです．このように走っているプログラムを止めるには，STOPキーを使います．もう1度，プログラムを走らせたいのであれば，

```
RUN ↵
```

と入力するか，あるいは，ファンクションキーのf・5と書かれたキーを入力します．PC－8801の電源を入れた状態の時には，ファンクションの5には，RUN ↵とキーボードを押したのと同じ働きをするように，定義されている訳です．また，STOPキーでプログラムの実行を止めたのですが，今回は，CONTという命令でのプログラムの再開はできません．今度は，STOPキーではなく，CTRL-Sで画面の表示を止めてみます．何度か，止めたり，また動かしたりやってみましょう．だんだん，STOPキー，CTRL-Sなどの使い方がわかってきたのではないでしょ

うか．それでは，次へ進みますので，プログラムの実行をSTOP（停止）させてください．もちろん，STOPキーを押しますね．

## サンプルプログラムを見る

さて，プログラムの実行は，止まっていますね．まだ，デモンストレーションをPC－8801がやっているのであれば，STOPキーを押して止めましょう．

では，これから，プログラムと呼ばれているものを見てみましょう．このプログラムは，付属のカセットテープに録音されていたもので，このプログラムによってあなたは，PC－8801のデモンストレーションを，楽しむことができた訳です．まずは，ホームクリアキーを押して，画面に表示されている文字を消しておきます．次にキーボードから，

```
LIST ↵
```

と入力します．どうです．画面の下から上へと間断なく表示されていくでしょう．画面の表示を一時止めておくには，何のキーを使いましたか．そうです．CTRL-Sでしたね．押してみます．表示が止まりましたね．もし，プログラムの表示がすべて終わってしまったのであれば，（その時には，最後の行にOkと表示されている）もう1度，

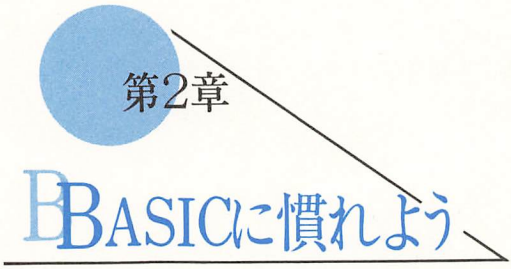
```
LIST ↵
```

とキーボードから入力します．そして，CTRL-Sで画面の表示を止めてみます．現在，あなたの見ているものが，プログラムと呼ばれるもので，このプログラムが，先程のデモンストレーションをやっていた訳です．何やら，数字と英字がごちゃごちゃと画面に表示されていますね．今は，プログラムとは，こんな風に，数字と英字がごちゃごちゃと入り交っているものだとでも覚えておいてください．そして，このプログラムというものを，これからあなたが，実際に作っていこうとしているのです．何だか気が遠くなってきましたか．心配しなくても，大丈夫ですよ．順序を追ってやっていけば，それほど難しい事ではありません．この第1章を，ここまで読んでこられたあなたなら，きっと出来ます．これから先も，がんばってやっていきましょう．

## まとめ

第1章を読んでみて，いかがでしたか．これから，自分にもやっていけると思えたのではないでしょうか．これから，第2章，第3章と読み進んでいくにつれ

て，覚える事もたくさん出てきますが，順序よく少しずつやっていけば，それほど難かしい事ではないことが，第１章を読んでみて，わかったのではありませんか．やはり，コンピュータを相手にするからといって，初めから構えてしまうのが，１番よくない事だと思います．話し相手が１人増えたとぐらいに考えていたら，どうでしょうか．それも，非常に論理にきびしい話し相手ですから，あなたが，間違った事を言ってきた時には，即座にエラーメッセージにより返答してきます．しかし，計算することに関しては，正確に短い時間でやってくれます．強い味方ではありませんか．そう思いませんか．


第2章
BASICに慣れよう

## “BASIC”って何？

PC-8801は，電源を入れると同時に，BASIC言語の使えるコンピュータに変身します．それではBASICとは何でしょうか．

それはちょうど，あなたとPC-8801の間に立ってあなたの言うことを，PC-8801に伝えてくる翻訳者のようなものです．あなたが「BASICさん」の理解できるような言語を使うことさえできれば，BASICはコンピュータが理解できるように指令を発して，うまくあなたの目的をかなえてくれるのです．ただ，普通の翻訳者とちがうのは，疲れたり，不平をいったりすることのないかわりに少々石頭で，あなたがあいまいな言い方をすると，理解できなくて〝エラー〟になったり，予期しないことを始めたりするのです……．ですからあなたの考えをコンピュータに伝えてくれれば，BASICでなくても構わない訳です．たとえば，FORTRAN言語とかCOBOL言語と呼ばれるものであってもよいのです．まだ他にもコンピュータの上で走る言語には，いろいろなものがあります．それぞれの言語にはそれぞれの特色があります．その中で比較的優しい言語が，BASICなのです．しかし，優しいと言っても，あなたが日本語の他に英語等を覚え始めた時のことを思い出してください．最初，多少戸惑いを覚えたのではありませんか．英語を理解するには外人と話をしてみるようなことで，英語に慣れることが理解を深める最短距離です．もし，あなたが今までに1度もコンピュータと接したことがなく，BASICがあなたにとって，初めてのコンピュータ言語であるとしたら，多分，大いに戸惑うことでしょう．しかし，慣れるということは，大変すばらしいことで，私達は，日常自然に日本語を話すのと同じように，自然にBASICが話せるようになってくるのです．そうなれば，もうしめたものです．その頃には，「そろそろ別の言語を勉強してみようか．」という具合になるかもしれませんが，1つでもコンピュータ言語を知っていると，他の言語もそれほど抵抗なく覚えることができます．また，あなたの考えた新しい言語をPC-8801の上で走らせることもできます．そういう意味で，今あなたは新しい世界に入って行こうとしているところです．まずは，このBASICを征服してしまいましょう．

BASIC言語の，他の言語と違う点に，会話型言語であることがあげられます．まさに，PC-8801と会話しながら作業していく訳です．あなたの目の前にあるキーボードから，PC-8801に話しかけることができます．また，PC-8801の方からあなたへは，ディスプレイ装置の画面を通して返答してきます．どうです．おもしろそうでしょう．しかし，まだあなたは，PC-8801と話しをすることはできません．それは，PC-8801の話せるBASIC言語の文法を知りませんし，BASIC言語の命

令も知りません．当然ですね．そこで，この文法と命令を，少しずつ覚えていこうという訳です．何も急ぐ必要はありません．あなたの時間の許す範囲でやっていけばよいのです．着実に一歩一歩！

## ダイレクトモードで使う

BASICの命令を覚えて，その命令を文法に従って，PC-8801にキーボードから入力すると，PC－8801はその通り実行してくれます．このように，すぐに実行させてみる使い方は，ダイレクトモード（直接実行形）と呼ばれ，コンピュータ本来の使い方とは多少違う使い方ですが，このダイレクトモードで使えることも，BASICの特長です．その使い方に慣れると，便利なものです．この章では，ダイレクトモードにおける使い方について書かれています．もう１つ，コンピュータ本来の使い方として，プログラムモード（間接実行型）と呼ばれるものがあります．このプログラムモードについては，第３章以降で書かれています．とりあえず，これからダイレクトモードでの使い方について，やっていきましょう．

これから先，いろいろとキーボードから入力をしてもらうことになりますが，私達日本人は，どうもこのキーボードとは余り縁がないので，キーボードから押したい文字を探すのに，かなり時間がかかってしまいますが，落ち着いて，１文字１文字，本書に書かれている通りに入力していきましょう．たとえば，もし押すキーを間違えてしまって，それが途中でわかったのなら，┘（リターンキー）と書かれているキーを押してしまいます．PC-8801からメッセージが返ってくるでしょうが，気にせずに，新しい行に，気分を入れかえて再び今度は間違えないように注意して入力してください．この時，PC-8801から返ってくるメッセージのことを，エラーメッセージと呼び，PC-8801には理解できないことを，あなたがキーボードから入力した訳です．早い話が，あなたが間違いをした訳です．別に間違えたくて，間違える訳ではありませんよね．しかし，あなたは正しいと思って入力したことに対しても，時としてPC-8801は，エラーメッセージを返してきます．その時には，素直に自分の間違いを認めてください．PC-8801を疑うのはやめましょう．きっと，自分の間違いに気付くはずです．それでは，これからあなたが次々とお目にかかるであろうエラーメッセージについて紹介しておきましょう．

### ?SN Error（Syntax error）

非常によくお目えにかかるメッセージの１つです．どういう時に表示されるかというと，そのほとんどがあなたのタイプミスです．キーを間違えて押した時に

顔を出します．すでに何度かあなたも，このメッセージをPC-8801から受け取ったのではありませんか．Syntaxとは，文法の事ですから，BASICの文法に合っていない時には当然このエラーメッセージが表示されます．もう1度，間違えている部分を直してからトライすれば，今度は大丈夫のはずです．

### ?FC Error（Illegal function call）

このエラーメッセージが出ると，少々やっかいです．というのは，まずは，どうしてエラーが起きたのかを調べなければなりません．このエラーの意味は，PC-8801にはできないことや，無理にあなたがやらせようとしているという意味です．その場所を必死になって探しましょう．このエラーメッセージは，プログラムモードで使っている時に出ることが多いのですが，たいていその場合には，プログラムが間違っています．プログラムをよく見直してみます．ダイレクトモードで，このエラーが起きた時には，実行した命令の書き方が間違っています．リファレンス・マニュアル等でもう1度その命令の書き方の確認をした上で，もう1度トライしましょう．

### ?/0 Error（Division by zero）

これは，見ればわかりますね．0で割り算をした場合に，このエラーは起こります．しかし，0で割る時に，分母だけが0（不定）と，分子，分母とも0（不能）の場合がありますが，この区別はしてくれませんので，そこまで知りたい時には，プログラムで工夫する必要があります．普通，そこまでやる必要はないと思いますが……．

### ? MO Errcr（Missing operand）

普通命令を書く時には，その命令に続けて，いくつかの数字を，（コンマ）で区切って書きます．もちろん命令だけで，その他に何も数字を指定しない命令もありますが，これらの数字の事を，以後パラメータと呼ぶことにします．覚えてください．そして，このエラーが起こるのは，そのパラメータの書き方が間違っている場合です．もう1度リファレンス・マニュアル等で，その命令の書き方（パラメータの指定の仕方）を調べた上でやり直しましょう．

以上，あなたがよくお目にかかるであろうエラーメッセージをいくつかあげましたが，まだ他にもたくさんのエラーメッセージはあります．もし，ここで書かれているエラー以外のメッセージが表示されたなら，迷わずリファレンスマニュアル等でその意味を調べて，少しずつ覚えていきましょう．

※カッコ内のフルスペルはディスクバージョンのメッセージです．

## 文字の色を変えてみよう

現在，あなたの見ている画面の文字の色は，何色をしていますか．もし，文字が白で表示されているのであれば，これからやることを楽しんでもらえると思います．それ以外の，グリーンディスプレイ装置をお持ちの方は，文字は緑色で表示されていますね．その方は，これからやることを同じように実行されても構いませんが，文字の色が，濃い緑になったり，あるいはうすい緑になったりするだけで，あまり楽しくはないかもしれません．

さて，これから何をやっていくのかというと，見出しを見ておわかりのように，表示される文字の色を変えてみようというものです．PC－8801は，この文字の色として合計8色のカラーを出すことができます．まずは，カラーを変える準備をしましょう．命令を1つ紹介します．CONSOLE（コンソール）命令といいます．何をするのか詳しい説明は，リファレンス・マニュアルを見てもらえばよいのですが，ここでは，画面を白黒モードからカラーモードに変えるために使います．では，白黒モードとか，カラーモードとかいうのは一体何でしょうか．多分，今あなたの見ている画面は，白黒モードだと思います．このモードでは，どう逆立ちしても，文字の色は変わりません．そこで，この白黒モードからカラーモードと呼ばれる状態へと，画面を変えてやります．このように，画面のモードとしては，白黒モードとカラーモードと呼ばれる2つの状態があるのです．さっそく，CONSOLE命令を使ってカラーモードにしてみましょう．次のように入力します．

```
CONSOLE 0,25,1,1 ↵
Ok
```

間違えないように．これで，いつでも文字の色を変えることができるようになりました．それでは，簡単にCONSOLE命令を説明してみます．まず，CONSOLEに続けて，4つの数字を指定できますが，3番目と4番目の数字に関しては，必ず0か1のどちらかです．

```
CONSOLE 0,25,1,2 ↵
Ok
```

エラーメッセージが返ってきました．

```
CONSOLE 0,25,1,-1 ↵
?FN Error
```

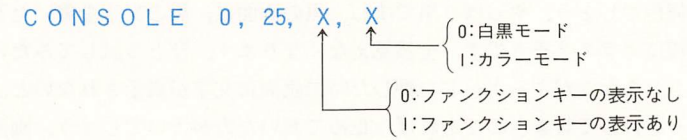
これでもだめです．では，次は，

```
CONSOLE 0,25,-1,1 ↵
?FN Error
```

何が実行されたのか，わかりましたか？よく画面を見ていてください．

```
CONSOLE 0,25,0,1 J
Ok
```

もうわかったでしょう．そうです，画面の１番下の行に，ファンクションキーの表示がされていますね．CONSOLE 命令の３番目の数字は，このファンクションキーの表示をするかしないかの選択ができるのです．ここを０にすれば，表示されなくなり，１にすれば逆に表示されるわけです．同じように，４番目の数字に０または１を指定すると，先程から話をしている．白黒モード，カラーモードの選択ができます．対応は，もうおわかりですね．０が白黒モードに，１がカラーモードにそれぞれ対応します．まとめると，




と，なります．では，１番目と２番目の数字は，何を指定しているのでしょうか．この説明はしないでおきます．あなたへの宿題とします．リファレニス・マニュアル等で，調べてみてください．そして，実際に使ってみてください．便利なことができるようになりますよ．ちなみに，この１番目と２番目の２つの数字をペアにして１つのことを指定します．

では，見出しにあるように，実際に文字の色を変えてみましょう．キーボードから，次のように入力します．１字１字，間違えないように！

```
CONSOLE 0,25,1,1 J
Ok
COLOR 7 J
Ok
```

何か文字をキーボードから入力してみます．文字の色は何色ですか．今までと同じ白ですね．では次です．

J（エラーメッセージを返してきても，続けて）

```
COLOR 4 J
Ok
```

どうです，色が変わったでしょう．緑色になったはずです．適当にキーを押して画面に文字を表示させて確めてください．このように，COLOR（カラー）に続けて数字を指定すると，文字の色を変えることができるのです．数字と色との対応は，次の通りです．

```
COLOR  0     : 黒
COLOR  1     : 青
COLOR  2     : 赤
COLOR  3     : 紫
COLOR  4     : 緑
COLOR  5     : 水色
COLOR  6     : 黄
COLOR  7     : 白
```

合計８色ありますが，０を指定すると色は黒ですね．あなたの見ている画面の地の色は何色でしょう．やっぱり黒ですね．黒の画面に，黒の文字を書いたとすると，闇夜にカラス？そうです，全然見えなくなります．ひとつ試してみたいのですが，もしあなたがキーボードを押した時に画面に文字が表示されないと，間違えてキーを押してしまうのであれば，止めておいた方がよいでしょう．画面を見なくても，キーボードが正しく押せるのなら，

↵（エラーメッセージを返してきても，続けて，）

```
COLOR 0 ↵
Ok
```

キーを２，３個押してみましょう．あなたには，画面に表示された文字が見えますか．見えないでしょう．しかし，黒で文字が書かれているのです．本当ですよ．やはり，キーボードを押した時には，画面に文字が表示される方が気持ちがいいものです．元に戻しましょう．

↵（エラーメッセージを返してきても，続けて，）

```
COLOR 7 ↵
Ok
```

間違えないように，１字１字キーを確認して押していきましょう．打ち間違いをしなければ，元のように，白で表示されるようになります．ならない時には，必ず打ち間違いをしていますから，何度か繰り返すうちに，元に戻るでしょう．黒以外の残りの色に関しては，あなたにお任せします．いったいどんな色なのか，実際に試しておいてください．

## 画面をクリアする

これから先，何度か「画面をクリアしてください．」という指定をあなたにすると思います．その時には，次の２通りの方法のどちらかを実行してください．こ

の，画面をクリアするとは，頂度，黒板消しで黒板の隅から隅まできれいに消すことと同じです．

### 最初の方法

CLR と書かれたキーをキーボードから探します．よーいドン！見つかりましたか？CLR とだけ書かれたキーは，実はキーボードにはないのです．ごめんなさい．本当は，HOME と上段に，CLR と下段に，1つのキーに上下二段に書かれたキーのことだったんです．見つからない方のために，場所はキーボードの右側にある数字の書かれたキーの集まり（テンキーと呼ばれている）の7のキーの上にあります．もう見つかりましたね．このキーを押してみます．一瞬にして画面はクリアされます．カーソルは，左上隅でチカチカしていますね．この方法だとキーを1回押すだけで済むので，非常に簡単です．

### もう一つの方法

今度は，キーボードから，

```
CLS J
Ok
```

と入力します．これでも，りっぱに黒板消しの役目は果せます．さっきの方法とちがう点は，キーを3回押さなければならないことと，画面の左上隅には，Ok と表示され，その次の行にカーソルがチカチカしていることです．どちらの方法でも構いませんが，両方共覚えておいてください．

その他にも，画面をクリアする手段としては，いくつかあります．これまでにも，何回か経験していませんか．あの CONSOLE 命令を実行した時に，白黒モードからカラーモードへ，あるいはその逆で，カラーモードから白黒モードへ，画面のモードを変えると自然に画面がクリアされます．覚えておくと役に立ちます．もう1つ，これと同じケースで，WIDTH（ワイズ）命令を実行しても画面をクリアすることがあります．初めて出てきた WIDTH 命令ですが，何をする命令かといえば，非常に簡単です．画面の桁数と行数の設定をするだけです．現在，あなたの見ている画面は，40桁の20行だと思います．つまり，横には最大40文字，縦には最大20行の，合計800文字分表示させることができるはずです．しかし，PC-8801 は，最大2000文字を一画面に表示させることが可能です．桁数が倍の80文字になります．それには，WIDTH 命令を使って，

```
WIDTH 80,25 J
Ok
```

どうです．桁が倍になって，今までの文字の大きさの半分の大きさで１つの文字が表示されているでしょう．見にくいというのではあれば，元の40桁に戻しましょう．

```
WIDTH 40,25 ↵
Ok
```

気が付きましたか？桁数を，40桁から80桁，あるいはその逆の80桁から40桁へ変える時には，画面がクリアされるでしょう．また，１つ注意があります．桁数と行数の指定は，任意に選べません．

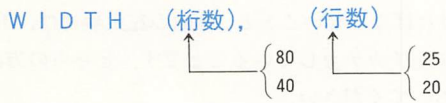
```
WIDTH 60,20 ↵
?FN Error
```

エラーメッセージが返ってきます．

```
WIDTH 40,15 ↵
?FN Error
```

同じように，エラーメッセージ．

実をいうと，桁数は，80桁または40桁のどちらか一方，行数も20行または25行のどちらか一方の指定しかできないのです．したがって，合計４種類の組合せ方がありますね．あなたの見やすい状態にしておきましょう．以上，WIDTH 命令について，まとめると，




と，いう訳です．

## ↵（リターンキー）その働きは？

今まで，↵と書かれている所では，あなたは，↵と書かれたキーを押していたと思いますが，では，この↵（リターンキー）は何のために押していたのでしょう．このことについて考えていきましょう．実は，これは非常に単純なことなのです．仮に，あなたとコンピュータの立場を逆転させてみましょう．いいですか，あなたがコンピュータに，コンピュータがあなたになるのです．そして，あなたの顔をしたコンピュータが，キーボードから，COLOR と入力してきました．もちろん，COLOR は何をする命令かあなたは知っています．コンピュータは先程の COLOR に続けて，スペースキーを１回ポンと打ち，しばらく考えた後に，０を押しました．あなたは，COLOR で０を指定すると，全く文字が見えなくなってしまうのを知っているので，必死に「０はやめた方がいいんじゃない．」とコンピュー

タに言いたいのですが，聞こえるはずはありません．そこで，あなたは覚えたての命令であるカラーを黒にしようとするのですが，何かの力で止められていて，実行できません．コンピュータの顔もCOLOR　0が実行されないので，イライラしているようですが，急に思い出したように，┘（リターンキー）を1回押しました．突然，今まであなたを押さえつけていた力が消え，あなたはカラーを黒に変えることができるようになりました．コンピュータは，画面に何も表示されなくなってしまったので，ブツブツと言っているようです．同時に，カラーを黒に変えた瞬間，またあなたは，何かの力を受けるようになりました……．

もう以前のあなたに戻ってくださって結構です．コンピュータも元の体に戻してあげましょう．この話の中であなたを押さえつけた力とは，一体何だったのでしょう．実は，これは┘（リターンキー）だったのです．正確に言うと，┘（リターンキー）が押されるまで，あなたは命令を実行できなかった訳です．しかし，よく考えてみると，この┘（リターンキー）があって初めて，コンピュータとのやりとりがうまくできるようになるのです．コンピュータ（本当のコンピュータですよ）は，┘（リターンキー）が押されるまで，何も実行せずに，押されて初めて，あなたからの命令を実行し始めるのです．つまり，あなたとコンピュータとのやりとりを，この┘（リターンキー）が区切りをつけているのです．

いままで，命令の最後には，必ず┘という記号がありましたが，これから先この┘という記号は省略します．つまり，特別な指定がない限り，

```
COLOR 7
```

と書かれていたならば，

```
COLOR 7 ┘
```

というのと，同じことだと考えてください．ですから，┘がないからといって，リターンキーを押さなくてもよいのではありません．しっかり押してください．約束です．

## スクリーンエディタ　（Part Ⅰ）

今まで，キーの打ち間違いをしたことのない方にとっては，これから書いていくことは余り役に立たないことです．しかし，人間いつ間違いをするかわかりません．読んでおいて損をしないと思います．

あなたは，いままでに←，→，↓，↑と書かれているキーが何度か目に入ったことでしょう．矢印が書いてあるくらいですから，何かの方向を示しているので

しょう．これは，カーソルの動く方向を表わしています．４つのキーの中から，好きなものを選んで押してみましょう．カーソルがその押されたキーの方向に動いたでしょう．この４つのキーを使うと自由にカーソルを動かせる訳です．では，１度画面をクリアしてください．やり方は２通り，お好きな方をどうぞ．次のように入力します．

```
CALAR 7
```

┘（リターンキー）はまだ押してはいけません．押してしまった人は，エラーメッセージのお世話になりましたね．もう１度やり直してください．それ以外の方は，今カーソルは７の後でチカチカしているはずです．CALAR を COLOR に直したいのです．どうしたらよいでしょう．カーソルを左へ動かします．←の書かれたキーを６回押して，Cの隣りのAにカーソルを合わせます．そして，キーボードから，Oを１回押します．どうです．AがOに変わって，カーソルが今度は，Lの所にきているでしょう．Oの所にあった先程のAはどこへ行ってしまったのでしょう．このように，カーソルを動かして，すでにある文字の上から，新しい文字を上塗りする（キーを押す）と，古い文字はなくなり，新しい文字が画面に現われます．もう１箇所AをOに直さなければいけませんね．やってみてください．まずカーソルを，→のキーを１回押して，Lの隣りにあるAに合わせてから，Oのキーを１回押します．今度は，カーソルの位置はRの所にきていますね．これで，めでたくエラーメッセージを出さずに済みそうです．┘（リターンキー）を押してみましょう．１つ気になることは，カーソルの位置ですね．┘（リターンキー）を押す前には，Rの所にありましたが，このカーソルを，７の後ろにもってこなくてはいけないようですが，このままで構わないのです．これは，┘（リターンキー）を押した時にエラーメッセージを出さないことからもわかりますね．キーをよく打ち間違える方にとって，この４つのキーは，魔法のキーです．どんどん活用しましょう．

## 打ち間違いに気付いたら

カーソルを動かして間違いを直す方法はすでにやりました．しかし，もっと単純な間違いもありますね．次のように，

```
COLL
```

┘（リターンキー）はまだ押してはいけません．COLOR と打とうとして，Lを２回押してしまいました．カーソルは後ろのLの次でチカチカしています．この

時，カーソルを左に１つ動かして間違えたＬの上からＯを打ち直しますか．もちろん，それでも構いませんが，ここで，１文字削除の術をあなたに授けましょう．術などと大げさに書きましたが，簡単なことです．

それには，キーを１つ探してもらわないといけません．前に登場した上段にHOME，下段にCLRと書かれたキーと同じように，上段にINS，下段にDELと書かれたキーを見つけてください．ありましたか．見つかったら押してみます．後ろのＬが消えて，その場所でカーソルが，チカチカしているでしょう．これで１文字（Ｌ）を削除できたのです．何度か繰り返すことで，何文字でも削除できます．ついでに，あと３回ほどこのキーを押してみます．COLが３文字分なくなりましたね．

次は，１文字挿入をやってみます．キーボードから，

```
COOR 7
```

と打ちます．↵（リターンキー）はまだ押さないで．見てわかるように，Ｌが抜けています．カーソルは７の後ろにあります．まず，カーソルを後ろのＯの所に持っていきます．そして，次に，SHIFTキーを押した状態で，先程の上段にINS，下段にDELと書かれたキーをポンと１回押します．えっ，何も変わらないって．ちょっと待ってください．それでは，Ｌのキーを１回押してみましょう．どうです，ＯとＯの間にＬが挿入されたでしょう．実は，この場合１文字挿入という言い方は，ちょっと違っています．現在の状態で，何文字でもＯとＯの間に挿入することができます．次けて，１２３４と４つのキーを押すと，

```
COL1234OR 7
```

となり，カーソルは依然としてＯにありますね．この挿入は，カーソルの位置を動かすかあるいは，もう１度SHIFTキーを押した状態で，先程の上段にINS，下段にDELと書かれたキーを押すまで続きます．逆に言えば，４つの矢印のキーのどれかを押して，カーソルの位置を変えるか，あるいは，もう１度先程と同じ要領で１文字挿入をするように，SHIFTキーを押した状態で上段にINS，下段にDELと書かれたキーを押すと，もうそれ以上挿入されなくなります．

カーソルの位置を動かす方でやってみましょう．左向きの矢印を１回押して，カーソルを，Ｏの隣りの４へ動かします．スペースキーを１回押します．どうです．４が消えてスペースになりましたね．挿入されなかった訳です．これで，カーソルは，次のＯの所にきていますから，先程やった１文字削除を４回やって，１２３を削除します．今度は，SHIFTキーは押す必要ありませんね．上段にINS，

下段に DEL と書かれたキーを 4 回押せばよいのです.

```
COLOR 7
```

と正しくなりましたか. 途中, 余計なことをしたので返って面倒うになりましたが…….

## PRINT命令を使う

あなたはすでにいくつかの命令を覚えていますね. さらにここで 1 つ, 新しい命令を追加します. この命令を覚えることで, あなたの目の前にあるコンピュータを電卓に変身させることができます. 電卓と言ってもあの四則演算 (+, −, ×, ÷)だけしかできないものではなく, 普通関数電卓と呼ばれるかなり本格的な電卓です. ではこの新しい命令について説明していきましょう. まずは次のようにタイプしてみます.

```
PRINT "HELLO"
HELLO
Ok
```

↵ (リターンキー) も忘れずに打ちましたか. 結果はどうなりましたか. ″(ダブルクォーテーションマーク) のキーの場所がわかりませんか. キーボードの左上の方に数字の 2 がありますね. SHIFT キーを押している状態にして, この 2 を打てば, ″が画面に出るはずです. 次の行にHELLO と出ましたか. エラーメッセージが返ってきた人はどこかを打ち間違えていますので, 打ち直しましょう. さてこの PRINT という命令は一体何を実行してくれたのでしょうか. 画面を見ればお解りですね. ″″で囲まれた中の文字を, 次の行にプリントしてくれた訳です. 本当にそうでしょうか. もう少し試してみましょう.

```
PRINT "GOOD MORNING"
GOOD MORNING
Ok
PRINT "GOOD EVENING"
GOOD EVENING
Ok
PRINT "GOOD BY"
GOOD BY
Ok
```

どうですか. 先程と同じように″″で囲まれている文字を次の行にプリントしてくれますね. ↵ (リターンキー) を忘れないで. 各行の終わりにはどれも必要ですよ. さて次はどうでしょう.

```
PRINT HELLO
```

〝〞で囲むのを忘れました．もう ┘(リターンキー) を打ってしまいましたか．試しに実行してみましょう．エラーメッセージが返ってくるかな．アレッ，何やら次の行に，

```
 0
Ok
```

とプリントされましたね．どうしたのでしょう．〝〞がなくても何かプリントされるようです．次のようにまたタイプしてみましょう．

```
PRINT 3
 3
Ok
```

次の行には何がプリントされましたか．3とプリントされているはずです．どういうことでしょうか．次はどうでしょう．

```
PRINT 15
 15
Ok
```

次の行には15とプリントされますね．このようにPRINTという命令は〝で囲んだ場合にはその囲まれた文字が次の行にプリントされますし，PRINTの後に数字を書いた場合，次の行にはその数字がプリントされるという2つの働きを持っています．でも待ってください．先程，

```
PRINT HELLO
```

という命令を実行させた時には，確か0がプリントされてきましたね．どうしてHELLOは0がプリントされたのでしょう．しかしこれについてはまだ説明しないでおきます．これは後から詳しく説明されています．お楽しみに．

## 電卓と比べると

```
PRINT 2
```

とタイプすると次の行には2がプリントされることは，すでにやりました．これは「2を次の行にプリントしなさい」と言う命令を実行しています．では，次の場合は何がプリントされるでしょう．

```
PRINT 1+2
 3
Ok
```

何がプリントされましたか．そうです．3がプリントされたでしょう．これは，何が実行されたのでしょう．言うまでもなく，「1＋2の計算結果を次の行にプリントしなさい．」という命令を実行した訳です．どうです．あなたはすでに，コンピュータを使って1＋2の計算をさせてその答えが3であることを知りました．えっ，コンピュータを使わなくてもそれぐらいわかるって．まあ，そう言わずに．では次はどうでしょう．

```
PRINT 123+456+789
 1368
Ok
```

答えはいくつになりましたか．そうです．1368です．えっ，まだタイプを打っている間に暗算した方が早いですって．わかりました．でも，もう少し式が複雑になったらどうでしょう．試しに計算させてみましょう．その前に通常私たちが四則演算をする時に使う記号としては，＋，－，×，÷を使います．そこでコンピュータでもこれと同じ記号を使いたいところですが，＋と－はそのまま使うとしても，×と÷はどうしましょうか．÷という記号をキーボードから探して御覧なさい．ありましたか．多分ないはずです．また×についても，X（エックス）という字と非常に紛らわしいですね．そこで，コンピュータでは，四則演算の記号として，＋，－，＊，／を使います．＊はアスタリスクと読み，かけ算を意味します．／はスラッシュと読み，わり算を意味します．わかりましたか．式の中に＊や／が出てきても，もう慌てる必要はありませんね．それではさっそく，

```
PRINT 123*456
 56088
Ok
```

答えはいくつになりましたか．56088が次の行にプリントされていますか．3桁どうしのかけ算だと暗算でやるよりは早いのではありませんか．まだタイプするのに時間がかかるって．そうですか．それでは1つ大変すばらしいことをこれからやってみせます．

```
? 123*456
 56088
Ok
```

まず，？のキーをキーボードから捜して打ちます．どこにあるのかわかりました

か．SHIFT キーを押しながら／が書いてあるキーを打てば？が画面に出ると思います．／のキーと言いましても，キーボードの右側の数字のキーの集まり(テンキーと呼びます）の中の／のキーではありませんよ．見つかりましたか．そしたら，次に先程の計算式を書きます．１２３＊４５６ですね．そして↵(リターンキー)です．びっくりしましたか．先程と同じように次の行には56088とプリントされています．種証しはこうなります．つまりＰＲＩＮＴと５つのキーを打つ変わりに，？と１つキーを打つことで代役を務めさせることができるのです．キーを打つ回数が４回少なくなって，早く答えが得られるようになったでしょう．でも，電卓なら計算式だけを押せば答えが出るって．そうですね．確かにこの程度の計算ならば，コンピュータは電卓に負けるかもしれません．しかし，あなたは電卓を使っていてこんな経験をしたことはありませんか．少し複雑で長い計算式を電卓だけで計算している時に，１回目で何らかの答えが出ます．これをそのまま答えとして使う人は，よっぽど注意して電卓のキーを押している人です．けれども，普通の人なら同じ計算をもう１度やってみると思います．これで１回目の計算結果と２回目の計算結果が同じであればよいのですが，もし，違っている時にはどちらの計算結果が正しいのか見当がつきません．しぶしぶもう１度同じ計算をしてみるでしょう．３回目の計算結果と１回目，２回目とを照らし会わせたうえで，やっぱり１回目が正しかったとか，２回目の方が正しいとか，こんな経験をしたことはありませんか．これは全て，電卓のキーをどういう順番で押したのかが，後からではわからないことに起因しています．つまり，後から確かめる手段がない訳です．これに対して，PC－8801 でやらせる場合はどうでしょう．計算結果と計算式が画面に残っている限り，後から何度でも確かめることができますね．これなどは，コンピュータ電卓の便利な点であるように思いますが，いかがでしょうか．

さて，それでは PC－8801 に演算をさせることができるようになったところで，実際にあなたが考えているのと同じように PC－8801 が計算をしてくれるかどうかについて考えていきましょう．その前に，先程コンピュータが四則演算に使う記号は，＋，－，＊，／の４つだということを書きました．実際には，この４つにさらにもう１つだけ加えることにします．それは＾という記号です．正式な呼び名はわかりませんが，これから先＾という記号は，〝やまがた〟とでも呼ぶことにします．この＾(やまがたと読む)の記号の意味は指数を表します．指数とは，あの $2^4$ とか $5^3$ という形で普段表わす数のことです．しかし PC－8801 では $2^4$ などという表示の仕方はせずに２＾４と表わします．さっそく試してみましょう．

```
PRINT 2*2*2*2,2^4
 16            16
Ok
? 2*2*2*2,2^4
 16            16
Ok
```

初めて出てきましたが，ＰＲＩＮＴ命令中ではコンマを使って，いくつかの数式を並べて書くことができます．結果はどうなりましたか．16が２つ並んでプリントされましたね．このように$2^4$などの指数を求める時には^を使うことで，キーを打つ回数を減らすことができます．それでは，あなたとＰＣ-8801が同じ式から同じ答えを出すことができるかどうか調べていってみましょう．

```
PRINT 10+10/2
 15
Ok
```

あなたの考えた答をＰＣ-8801はプリントしてくれましたか．考えられる答えとしては15と10ですね．もちろんＰＣ-8801は15という答えを返してきたでしょう．それではもう１つ，

```
PRINT 10+10/2,(10+10)/2
 15            10
Ok
```

前半は先程と同じ式で，当然答えは15となりますね．では，15の後にはいくつがプリントされましたか．そうです．10がプリントされているでしょう．どうです．考え方としては私たちが普段計算しているのと同じようにＰＣ-8801も計算してくれるようです．それではもう１つ例をあげます．

```
PRINT -10+20,-(10+20)
 10           -30
Ok
```

あなたの考えた答えをプリントしてくれましたか．次は少し難しくなります．

```
PRINT 2^3^4
 4096
Ok
```

これは，私たちが普段使う数式で書くと$2^{3^4}$とはならないのです．気を付けてください．$2^{3^4}$などとはＰＣ-8801は計算してくれませんよ．これは$2^{3\times4}$つまり，$2^{81}$の意味になります．ですから，$2^{3^4}$をPC-8801に計算させたい時には，かっこを使って，

```
PRINT 2^(3^4)
```

と，書かなければなりません．くれぐれも気を付けてくださいね．応用問題として１つ掲げておきます．

```
PRINT 2^2^2^2
```

答えとしてはそれほど大きな数ではありません．イチ，キュー，ヨン，パー，イチロク，……ムニャ，ムニャ……．

## A＝A＋1 !?　（代入文）

あなたがPRINT命令を理解してくれたおかげで，やっとコンピュータをコンピュータらしく使う事ができるようになってきました．

コンピュータは私たちと違って同時にたくさんの別々な数字を覚えていてくれます．最近の電卓はメモリという機能をほとんどが備えています．あなたも使ったことがあるでしょう，あのメモリの意味のMなどと書かれているキーのことです．あのキーを使った経験がある方なら，あれは数値の一時あずかりのような機能を持っていることを知っていますね．つまり紙とエンピツがない時等，あのキーを押しておくことで，別の計算をした後からでも，先程Mのキーを押した時点での数値を簡単に知ることができます．しかし一時あずけておくことのできる数値はたいがい１つに限られていますね．つまり電卓のメモリ機能は，同時にいくつもの数値を覚えていることができない訳です．コンピュータは，同時にいくつもの数値を覚えていることができます．けれどもここで１つ問題が出てきます．覚えている数値が１つだけならば，今覚えている数値はいくつですかと聞かれた時に迷うことなく答えることができます．しかし，いくつもの数値を覚えているとしたらば，同じ質問をされた時にどのように答えたらよいでしょうか．覚えている数値を片っぱしから教えてあげますか．これは先程と同じ質問の仕方をしたのではだめだということです．つまりいくつかあるメモリの中から１つを指定して，このメモリに「入っている数値はいくつですか．」というように聞かない限り日が暮れてしまいます．そこでコンピュータは，このメモリとかあのメモリでは何が何だかさっぱりわかりませんので，変数名という名前を使ってメモリの区別をしているのです．ちょっと前の話になりますが，あなたがPRINT命令を覚えたての頃にこんなことをしたのを思い出してください．

```
PRINT HELLO
0
Ok
```

結果は次の行に０がプリントされていましたね．どうです．わかってきたでし

ょう．つまりあの時，HELLO という変数名のついたメモリには 0 という数値が入っていた訳です．現在は，どうでしょうか．

```
PRINT HELLO
 0
Ok
```

やはり 0 がはいっているようです．それでは，あなたが電卓のMキーを押すのと同じようにして，この HELLO という変数名の付いたメモリに何か値を入れてみましょう．どのようにするかと言いますと，

```
HELLO=3
Ok
```

とします．この＝の記号は本来の←という意味からすれば適当ではありません．しかし，画面に←の記号が出せるキーがあなたのキーボードにありますか．あのスクリーンエディットの時に使う←のキーはカーソルを左方向に動かすだけで，画面に←の記号は出てきませんね．そんなことから←の記号の変わりに＝を使うことになっています．ここで非常に注意すべき事が＝を使うことによって生まれてくるのです．それは，先程 HELLO に 3 を代入しましたから，HELLO の現在の値は 3 になっているはずです．そこで次のようにタイプします．

```
HELLO=HELLO+1
Ok
```

何とこんな式が成立するはずはありません．しかし，コンピュータは無表情に Ok をプリントしてカーソルをチカチカさせているでしょう．確かにこの式の左辺と右辺は等しくはありません．当然ですね．けれども実はこの式は等しいとか等しくないとかを言っているのではなく，先程書いたように右辺の値を左辺にある変数に代入するだけの意味しか持っていないのです．この場合，HELLO にはあらかじめ 3 を代入してありましたから，

$$
\begin{aligned}
(\text{右辺}) &= \text{HELLO} + 1 \\
&= 3 + 1 \\
&= 4
\end{aligned}
$$

右辺は 4 であることがわかります．この右辺を左辺の変数 HELLO に代入する訳ですから HELLO＝HELLO＋1 が実行されると HELLO は 3 から 4 に変わるはずです．試しに HELLO の値をプリントしてみます．

```
PRINT HELLO
 4
Ok
```

確かに4になっていますね．この代入文の左辺には，必ず1つの変数がきます．右辺には変数名や計算式を書くことができます．つまり，代入は常に右辺から左辺に向かって実行されるということです．したがってHELLO＋1＝HELLOとタイプしてもコンピュータは理解できませんから何らかのエラーメッセージを返してきます．

また，次に問題になるのは，変数名の付け方です．リファレンス・マニュアルに，ていねいな説明がされていますので詳しい説明はそちらに任せるとして，一般的な注意だけしておきます．理想としては，変数名として，どんな名前でも使えるとよいのですが，PC－8801では，ある程度制限があります．これは，BASIC言語自体に起因していることですが，変数名として，たとえば，WIDTHというような名前は使えません．

```
PRINT WIDTH
?SN Error
```

もちろん，エラーメッセージが返ってきます．つまり，WIDTHというのは，BASIC言語がすでに私達よりも先に桁数と行数の設定をする時のために，命令として予約してあります．ですから使うことはできません．その他にも，すでに予約してある名前というのは，いろいろあります．これらを予約語と呼び，変数名を付ける場合には，この予約語以外の名前にしなければいけません．また，変数名というからには，何かしら値が代入される訳ですから，ただ，A，B，C等というように，漠然とした名前よりは，一目見て何に使っているのかわかるような名前にしておく方がよいでしょう．たとえば，年齢が代入される変数には，NENREI等というような変数名を付けておけば，一目見てすぐにそれが何に使われているのか，わかりますね．さらに，変数名の長さとしては，最大40文字まで許されていますが，長くても10文字前後にしておいた方がよいでしょう．

```
A123456789=9
Ok
PRINT A123456788,A123456789
 0             9
Ok
```

とすると，最後の10文字目が違うだけで，全然別の変数名と見られてしまいます．長くなればなるほど，このような間違いも増えるでしょう．40文字の長さの変数名といえば，頂度画面が40桁の時には，その1行分の長さになりますね．考えるだけでもゾッとしませんか．

## まとめ

この章では，ある程度BASIC言語に慣れていただきましたが，まだまだBASIC言語の門をくぐったに過ません．今までのような使い方をダイレクトモードでコンピュータを使うと呼んでいる訳です．しかし，門をくぐらないと先へは進めないのです．覚えることがたくさん出てきて大変ですが，ここで第2章で出てきた命令のおさらいをしておきます．参考にしてください．

CONSOLE　：画面のモードを白黒モードからカラーモードへ，あるいはその逆のカラーモードから白黒モードへ変更します．ファンクションキーの表示をしたりしなかったりするのにも使いました．もう1つの働きは，あなたが一番よく知っていますね．スクロールする行の指定をするのに使います．詳しくは，リファレンス・マニュアルをどうぞ．

WIDTH　：画面の桁数の指定をするのに使いました．

COLOR　：文字の色を変えるのに使いました．これ以外の働きもあります．

CLS　：画面をクリアするのに使いました．

PRINT　：画面にプリントする場合に使います．〝〟で囲めば，その囲まれた文字を，数式を書けば，演算結果を，次の行にプリントすることができます．

その他，特殊キーの使い方もやりました．

HOME・CLR キー　：画面をクリアするのに使いました．

↵（リターンキー）　：あなたが，コンピュータに化けさせられましたね．

←→↓→キー　：カーソルを動かすキーです．

INS・DEL キー　：一文字削除，挿入に使いました．

## フロッピーディスク装置をお持ちのあなたへ

PC－8801にフロッピーディスクがつながっている方を対象にして，これから話を進めていきますので，フロッピーディスクをお持ちでない方は，当然実行できない命令が出てきます．無理やり実行させようとしても，無駄です．あきらめましょう．ですから，読み飛ばされても構いませんが，フロッピーディスクについて全然知らない方は読みとばさずに読んでみてください．きっと，フロッピー

ディスクの便利さにびっくりされて欲しくなることでしょう．

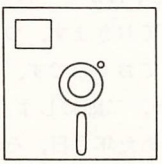
さてこれまで，カセットテープレコーダを使って，デモンストレーションプログラムの読み込みをしていたのですが，フロッピーディスクがつながっているのであれば，わざわざ時間のかかるカセットを使わずに，フロッピーディスクを使ってデモンストレーションプログラムを読むことができます．

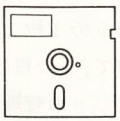
まずは，付属のマニュアルに従って，正しく，フロッピーディスクを使える状態にします．ここまでは，確実にやってください．これがうまくできれば，後は，カセットテープレコーダを使うのと同じ要領です．しかし，カセットテープレコーダを使ってやるのと比べると，極端に短い時間で，同じ事ができるようになります．何よりもこのフロッピーディスクを使うメリットは，かかる時間の短縮にあります．フロッピーディスクを使ったことのない方がカセットテープレコーダだけでやっている時には，カセットテープレコーダの便利さと早さに感激していると思いますが，一旦，フロッピーディスクを使ってしまうと，もうカセットテープレコーダを使う気がしなくなっていまいます．そのくらいフロッピーディスクとは便利なものです．しかし，その便利なフロッピーディスクも，使う上での注意があります．それを見ていきましょう．

## ディスケットは非常にデリケート

カセットテープレコーダを使う時には，媒体（メディア）としてカセットテープがありますね．つまり，実際に信号が記録されるのはカセットテープです．これと同じようにフロッピーディスクの場合にも，媒体（メディア）として，フロッピーディスケット（以後ディスケットと略す）があります．このディスケットに信号が記録される訳です．

カセットテープのテープ自体に手を触れて，テープ面上を汚してはいけないことは，誰でも知っています．同じように，ディスケットにも手を触れてはいけない部分があります．いままでフロッピーディスクをお持ちの方は，何度かこのディスケットに触れる機会があったことと思いますが，少し心配です．手許にディスケットがあるならば，ちょっと出してみてください．手許にもなくこれまでにも１度も見たことがないのであれば，図を見てください．


（8インチ）


（5インチ）

まず，ディスケットを自分の前に置きます．一応文字の書かれているシールを表にして文字が正常に読める向きにします．ですから，右側の上に逆コの字型の切れ込みが入っていますね．中心には孔が開いています．その下にも楕円の孔がありますね．中心の右下には，小さな丸い孔があります．外側から，フロッピーディスク本体に触われる場所というのが何箇所かある訳です．当然このフロッピーディスク本体に手を触れてはいけません．ディスケットを持つ時には，これらの孔を避けて，上部に貼られたシールの部分を持つようにします．間違っても楕内形の部分から，フロッピーディスク本体には触わらないように気を付けてください．

では，ディスケットを持ってみましょう．気を付けて．シールの部分に指をおいて．さすがにディスケットというだけあって，グニャグニャしています．容易に折れてしまいそうですが，決して曲げたりして遊んではいけませんよ．それから上に重たい物等を乗せておくと，簡単に変形してしまいます．わかりますね．ディスケットは大切に取扱ってあげてください．また，あなたの目には見えませんが，磁石等の磁気のある物の近くにディスケットを置いてはいけません．理由はわかりますね．

以上，“べからず集” になりましたが，このようにディスケットは非常にデリケートなものなのです．あなたの手で守ってあげましょう．

さて，何度かシールという言葉が出てきましたが，もう少し詳しく説明すると，ディスケットには，合計３枚のシールを貼ることができます．実際には，シールとは言わずにラベルと呼ばれています．

①標識ラベル
②ジョブ用カラーラベル
③ライトプロテクトラベル

これらの３つのラベルの正式な呼び方ですが，あなたの手で自由に貼ったりはがしたりできるのは，②と③の２つです．①はディスケットを買った時からすでに貼られていて，そのディスケットの製造メーカー等の名前が入っています．あなたのディスケットにも貼られているでしょう．そう，その左上の隅のラベルの事です．無理にはがせばはがれないことはないのですが，はがしてはいけません．このラベルには，そのディスケット固有の情報を書いておきます．つまり，あなたの持っているディスケットの１枚１枚に名前を付けておくのです．名前と言っても，人と同じ名前を付けて，「今日は一郎を使おうか，二郎にしようか．」ではありません．そのディスケットの管理番号や，使い始めた年月日，その他書き込

んでおけば，このラベルを見ただけでそのディスケットの生い立ちを知ることができますね．あなたも１つ工夫してみてはいかがでしょうか．なお，ラベルに書き込む時の注意を１つだけ．それは，ボールペンや鉛筆等を使って，ディスケットに貼られているラベルの上から，きつく書き込むと，当然ディスケット本体が傷つく可能性があります．必ず，サインペン等を使って，軽く書くようにしてください．

それでは，次に残りの２つのラベルについて，②のジョブ用カラーラベルとは，先程の①の標識ラベルの隣りに貼りつけるラベルのことです．ディスケットを買うと一緒に付いてくるはずです．名前の通り，このラベルには，そのディスケットは何に使われているのか見分けをつけるために貼っておきます．このラベルは自由に貼ったりはがしたりできますから，有効に使ってください．もちろんディスケットに貼りつける前に，ラベルの記入をしておきます．また，今まで貼ってあったラベルを新しいラベルに貼り変える時に，うまく貼がれないからと言って，上から新しいラベルを重ねて貼ってはいけません．横着せずに，きれいに貼がしてから新しいラベルを貼りましょう．

残るは，③のライトプロテクトラベルですね．これは，カセットテープと比べてみるとすぐにわかります．もしあなたが，カセットテープにあなたの好きな音楽等を録音して，もうこれ以上録音しない時に，消去される心配をなくすために，カセットテープの面の反対側にある爪を折っておくでしょう．最近では，爪を折るのではなく，プラスチック製のコの字型のピンをはさむのもありますが，③のライトプロテクトラベルとは，まさにこれと同じ働きをします．このラベルを，ディスケットの逆コの字型の切れ込みの部分に貼っておくと，このディスケットに書き込もうとした時には．得意のエラーメッセージで，「このディスケットには．書き込めませんよ．」とＰＣ-8801は教えてくれます，書き込んでも大丈夫なものであればディステットからこのラベルを貼がした後で．もう一度，書き込んでやれば．今度は書き込んでくれますですから．大事なディステットには，不用意な書き込みから守ることができます．大いに活用しましょう．

## ロード，セーブの仕方

あなたのフロッピーディスクは，１度に何枚のディスケットを使えますか．多分，２枚もしくは４枚のはずです．ディスケットを入れられる部分が，２箇所あるいは４箇所ありますね．カセットテープレコーダを使っている時には，対象とするカセットテープは１つだけでしたから，どのカセットテープということを指定しなくても済みました．しかし，フロッピーディスクには，最低２枚のディス

ケットが対象となりますから，どちらのディスケットを対象とするかを指定しなければなりません．これには，ドライブナンバーというものを使います．カセットテープレコーダの場合には，

```
LOAD "CAS1:DEMO"
```

これで，DEMOというプログラムをカセットテープから読むことができました．フロッピーディスクでは，

```
LOAD "1:DEMO"
```

となります．この1というのが，ドライブナンバーです．フロッピーディスクにも，このドライブナンバーが書かれていますね．どちらがドライブ1，ドライブ2なのかわかりますね．ですから，ドライブ1にディスケットが入っているのであれば，先程のように，

```
LOAD "1:DEMO"
```

と指定します．ドライブ2にディスケットが入っているのであれば，当然，

```
LOAD "2:DEMO"
```

となります．では，ドライブナンバーを忘れて，

```
LOAD "DEMO"
```

としてしまったらどうなるでしょう．エラーメッセージが出るでしょうか？いいえ，これは，

```
LOAD "1:DEMO"
```

と同じ意味になり，ドライブ1が指定されます．ドライブナンバーを省略すると，必ずドライブ1が指定されてしまいます．ですから，ドライブ2にプログラムをセーブしようとして，

```
SAVE "DR-2"
```

とすると，ドライブ1にセーブされてしまいますから，気を付けてください．
以上，ドライブナンバーの指定の仕方がわかれば，本書で，

```
LOAD "CAS1:------"
SAVE "CAS1:------"
```

となっている部分を，そのまま，

```
LOAD "1:------"
SAVE "1:------"
LOAD "------"
SAVE "------"
LOAD "2:------"
SAVE "2:------"
```

と変えるだけで，同じようにカセットテープレコーダの場合よりも短時間で実行することができます．多少，優越感に浸れることでしょう．

## ディスケットのクラッシュに備える

フロッピーディスクに付属のマニュアルには，フロッピーディスクをPC-8801が使える状態にした後で，まず，システムディスクのコピーをしています．システムディスクとは，フロッピーディスクをPC-8801が使えるようにするための大切な情報が記録されているディスケットの事です．では，どうして，このシステムディスクのコピーを最初にするのでしょう．それは，フロッピーディスクを使う上での１番の不安は，ディスケットのクラッシュにあるからです．クラッシュという言葉は，カーレース等でよく使われますが，何も，ディスケットの外観がメチャクチャになるのではなく，ディスケットの内部，つまり記録されているデータがメチャクチャになってしまう状態を言います．では，どういう時に起こるかというと，まず，あなたの意志に反して，プログラムが暴走した時とか，停電で一斉に電源が切れたとか，不用意にディスケットに磁石等の磁気の強い物を近づけてしまったとか，いろいろあります．もちろん，必ずクラッシュするとは言えませんが，常にディスケットにはクラッシュが付いてまわっています．正しく使っている分には，めったに起こることはありません．しかし，もし起こった時には，１枚のディスケットにはかなりの容量のデータを記録できるために，一瞬にしてそれらの大量のデータを失うことになります．そこで，先程のシステムディスクのようにそのディスケットのコピーをもう１枚の別のディスケットにとっておき，元のオリジナルのディスケットは，大切にしまっておき，コピーしたディスケットを使うようにします．コピーしたディスケットをクラッシュさせる分には，それほど大きな被害はないでしょう．またオリジナルのディスケットからコピーを取れば済みます．しかし，コピーしたディスケットに，大事なプログラムが入っていたりすると，当然クラッシュと同時に，その大事なプログラムも消滅してしまいますね．ですから，システムディスケットに限らず，大事なプログラム等は，別のディスケットにコピーをとっておき，いつクラッシュしても

いいように準備しておくのです．その準備は無駄になることかもしれませんが，その時にはディスケットがクラッシュしなかったのですから，よかったと思っていればよいのです．また，コピーがあるということで，幾分心配がなくなりますね．

## まとめ

以上，フロッピーディスクを使う場合のことについて書いてきましたが，まだフロッピーディスクをお持ちでない方は，欲しくなったでしょうか．フロッピーディスクを使うともっと便利なことがたくさんあります．全てについて触れている訳ではないので，「何だ，フロッピーディスクはカセットに比べて，プログラムの読み書きが早いだけじゃないか．」と，思われた方もいると思いますが，まだまだフロッピーディスクは，こんなものではないということを付け加えておきます．実際に，フロッピーディスクをお持ちの方は，付属のマニュアルをよく読んで，１日も早く使い方に慣れることです．初めのうちは，その早さや，記録容量の大きさに感激することでしょう．しかし，次第にその感激も，あたりまえとなり，逆に，「もっと早くならないのか」とか，「記録容量がもう少し欲しいなあ．」とか思えてくるかもしれませんが，それはあなたのわがままです．でも，そう思うようになっているあなたは，きっとフロッピーディスクの隅々まで使いこなしていることでしょう．

# 第3章

# BASICプログラミングの初歩

## “プログラム”って何？

プログラムって，一体どんなものなのでしょう．また，何故必要なのでしょう．とりあえず，これからいくつかのことを，PC－8801 に実行させてみます．間違えないように，タイプしていってください．

```
NEW
```

PC－8801 から Ok が返ってきましたか．もちろん，┘（リターンキー）が必要ですよ．忘れてはいませんね．この命令は，後から詳しく説明します．次に実際にの中に，どんなプログラムが入っているのか見てみましょう．

```
LIST
```

えっ，ただ Ok が返ってくるだけですって．そうなんです．先程 NEW という命令を実行したので，PC－8801 の中のプログラムは，全てなくなりました．でも，PC－8801 との話は前と変わらずにできますね．それでは，次です．

```
RUN
```

この命令は，プログラムの実行をする場合に使います．どうなりましたか．また Ok が画面に出るだけですね．PC－8801 は，RUN という命令をあなたから受けたので，プログラムの実行をしようとしたのですが，現在 PC－8801 の中にはプログラムはありませんから，実際には何もせずに，Ok を返してきたのです．余計わからなくなったなんて言わないでください．つまり，PC－8801 はプログラムがない状態では，電卓に大きな画面を付けたようなものなのです．PC－8801 をコンピュータらしく使うためには，このプログラムと呼ばれるものを PC－8801 の中に入れてあげなければなりません．ではやってみましょう．

```
10 PRINT "PROGRAM"
```

┘（リターンキー）も必要ですよ．次の行に PROGRAM とプリントされましたか．されていないはずです．これでは本当に PC－8801 の中に入ったのかわかりませんので，試してみましょう．画面を一度すべてクリアします．もうやり方はわかっていますね．そこで，次に，

```
LIST
10 PRINT "PROGRAM"
```

とタイプします．次の行には，先程あなたがタイプした一行が現われるはずです．

いかがですか．では実行してみます．

```
RUN
PROGRAM
```

どうです．次の行にPROGRAMとプリントされたでしょう．もう一度，

```
RUN
PROGRAM
```

前と同じようにPROGRAMとプリントされましたか．これであなたは，何度でもRUN命令を使って，次の行にPROGRAMとプリントさせることができます．つまり，言いかえれば先程あなたがタイプした一行は，次の行にPROGRAMとプリントするプログラムだったわけです．

## “ソフトウェア”＝“プログラム”??

ソフトウェアという言葉を，これまでに聞いたことはありませんか．コンピュータの世界では，よく登場する言葉です．また，このソフトウェアと対比させて，ハードウェアという言葉もよく登場します．ソフトとハード，コンタクトレンズみたいですね．このハードウェアとは，今あなたの目の前に置かれているコンピュータのことだと考えてください．そして，ソフトウェアとは，そのコンピュータをあなたの意志どおりに動かすためのプログラムのことだと考えてください．よくわからないって．そうですか．それでは，もう少し具体的に．コンピュータを作り上げている，あなたの手で触れることのできる部分をハードウェア，そのままの形ではあなたの手で，決して触れることはできませんが，コンピュータの中に入れることのできるプログラムのことを，ソフトウェアと呼びます．従って，どちらか一方があれば，よいというわけではありません．このハードウェアとソフトウェアとがあって，初めてコンピュータとして意味をもつようになります．

## 今までとはここが違う

ここまででであなたは，PC－8801を電卓と同じように使う方法を知りました．でも，これからは違います．やっと，PC－8801をコンピュータらしく使うことができるようになるのです．そのためには，行番号という概念を覚えなければなりません．先程，PROGRAMと次の行にプリントするプログラムをタイプした時に，行の初めに10とタイプしたことを覚えていますか．この10のことを行番号といいます．つまり，コンピュータといえども，たくさんの命令を覚える時には，この行番号をたよりにして覚えていくのです．ですから，電卓のような使い方をしている時には必要なかったわけです．また，PC－8801は，この行番号が行の最初に

タイプされると，「あっ，この行はプログラムの一部だから，すぐには実行しないで，RUN 命令がタイプされた時に実行すればいいんだな．」と考えて，PC－8801の中にしまっておいてくれます．試してみますか．

```
10 PRINT "LINE 10"
20 PRINT "LINE 20",
30 PRINT "LINE 30"
```

それぞれの行には，必ず ┘（リターンキー）を押すことを忘れずに．

確かに実行はしてくれませんね．では，一度画面をクリアして．

```
LIST
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 PRINT "LINE 30"
```

覚えてくれていますね．実行します．

```
RUN
LINE 10
LINE 20
LINE 30
```

うまく実行してくれましたか．さらに，

```
15 PRINT "LINE 15"
```

とタイプして，

```
LIST
10 PRINT "LINE 10"
15 PRINT "LINE 15"
20 PRINT "LINE 20"
30 PRINT "LINE 30"
```

とすると，今タイプした行が，10行と20行の間に挿入されています．PC－8801は，このように行番号の小さい順に，一行一行覚えていくのです．ついでに実行してみましょう．

```
RUN
LINE 10
LINE 15
LINE 20
LINE 30
```

確かに，先程の15行が10行と20行の間に入ったようです．このように，実行する場合もPC－8801は，行番号の小さい順に実行していきます．さて，それでは，

この行番号にはどんな数が使えるのか疑問になります．お答えします．行番号として使える数は，1から65529までの整数に限られます．しかし，1から初めて，1刻みに行番号を付けていくことは，めったにしません．何故なら先程，10行と20行の間に15行を挿入したように，ある行とその次の行の間に一行挿入したい場合が，プログラムを作っていくうちに，度々起ってきますが，その時に，1刻みで行番号を付けてしまっていると挿入できなくなるためです．ですから，あらかじめ挿入が起こることを考えて，行番号の間隔はとっておくのです．では，ここでミステリーを一つ．前にPROGRAMと次の行にプリントするプログラムはどこへいってしまったのでしょう．LISTをとって確かめてください．ありましたか．見つからないはずです．というのは，あの時使った行番号は10だったはずです．現在の10行には，LINE　10とプリントする命令が入っていますね．つまり，行番号がどちらも10行であったために，先程の10行は，今回新しく入れた10行と置きかわってしまったのです．それでは手品を一つ．

```
15
```

何度も書くようですが，↵（リターンキー）も必要ですよ．

```
LIST
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 PRINT "LINE 30"
```

あれっ，15行がなくなりました．

```
RUN
LINE 10
LINE 20
LINE 30
```

実行しても15行は見あたりません．行番号だけ書いて↵(リターンキー）を押すとその行が削除されることがわかりましたか．このように，プログラムを作っていくうえで，行番号の概念は非常に大切なものです．よく覚えておいてくださいね．しかし，コンピュータがどのようにしてこの行番号と命令をたくさん覚えていられるのかなどと考えてはいけません．第一，夜眠れなくなります．

## プログラミングのお話あれこれ

コンピュータに，何か仕事をさせる場合，プログラムが必要であることが，次第にわかってきたことと思います．このプログラムを作る作業をプログラミング

と呼びます．そこでこのプログラミングについて，実際にはどのように進めていったらよいのか，お話ししておきましょう．

あなたの身の回りで問題が生じたとします．あなたはすぐにその問題を，コンピュータを使って解決しようと思いますか．あなたが，プログラミングについての知識をそれほど持っていないとしたら，コンピュータを使おうとは思わないでしょう．しかし，これから先プログラミングが次第にわかってくると，即座にコンピュータと結びつけるようになります．でも，ちょっと待ってください．もう一度その問題を検討してみて，本当にコンピュータを使うべきかを，よく考えてほしいのです．コンピュータを使うことで，問題をより複雑にしてしまうこともあるかもしれません，また，コンピュータを使うとなると，プログラミングをするという多大な手間と時間が必要となってきます．ですから，コンピュータを使う時には，問題の分析を十分に行ったうえで，プログラミングしていくべきです．しかし，このことは，非常に難しいことですから，今すぐにという訳にはいかないでしょう．あなたが，これから先プログラミングの経験を通して，あなた自身で身につけていってください．忠告までに．

さあ，プログラミング．

いや，もう一度問題分析！

## ロード・ろーど・LOAD

LOAD命令は，すでに登場しました．覚えていますか．そうです．何やらおかしな音の録音されている，カセットテープを使って，コンピュータに聞かせるときに，この命令を使いました．あの時は，ただ教えられるままに，この命令をタイプしていきましたね．それでは，ここでこの命令の使い方を説明していきましょう．とりあえず，もう一度，付属のカセットテープに録音されているデモンストレーションプログラムを，PC－8801にロードしてみます．そうなんです．あのカセットテープには，プログラムが録音されていたのです．信じられませんか．そして，ロードするというのは，PC－8801の中に，カセットテープ等に記録されているプログラムを入れてやる命令ということになります．そこで，先程のように「プログラムをPC－8801にロードする」というような使い方をする訳です．さっそく，やってみましょう．

```
LOAD "CAS1:DEMO"
```

うまく見つけてくれましたか．やり方のわからない人は，もう一度第1章の終わりを御覧ください．さて，では，コンピュータが，デモンストレーションプロ

グラムを読を込んでいる間に，〝CAS1 ： DEMO〟について説明しましょう．LOADという命令の後ろに，なぜこれが必要なのでしょうか．もうおわかりですね．PC－8801に対してロードしろと命令しても，PC－8801は，「どこから，どういうプログラムをロードすればよいのでしょう．」と当然，聞きかえしてきます．この，どこからというのと，どういうプログラムをというのに対して，LOAD命令の後ろで指定するわけです．それも，〝(ダブルコーティション)で囲むことを，お忘れなく．どこからというPC－8801の質問に対して，あなたは，カセットテープかディスク装置かの二通りの返答の仕方がありますね．そこで，カセットテープからであれば，CAS1 ：を指定します．ですから，先程の場合これを使った訳です．次に，どういうプログラムかという質問に対しては，どうしましょう．これには，プログラムをファイル名というものを使って区別します．詳しい説明は，もう少し後でしますが，簡単に説明すると，プログラムをカセットテープ等に記録する場合，プログラムに名前をつけて記憶させておけば，大変便利ですね．また，ロードする時のことを考えて，プログラムをファイル名というもので，一緒に記録しておくのです．そうすれば，後でそのプログラムをロードしようと思った時に，そのプログラムファイル名を頼りにすることができます．ですから，このプログラムファイル名には，そのプログラムに関係のある名前を付けておくと，後から頭を悩まさずに済みます．いい名付け親になってくださいね．今回のこのデモンストレーションプログラムには，DEMOというプログラムファイル名がついている訳です．わかりやすいでしょう．そろそろ，PC－8801もロードを完了したでしょうか．

## プログラムを見る

デモンストレーションプログラムのロードが完了して，PC－8801がOkのメッセージを返してきたら，あなたは，何をしますか．すぐにプログラムの実行をするためのRUN命令をタイプしますか．それとも，LIST命令を使ってプログラムを見てみますか．以前にこのプログラムは実行したことがありますね．それでは，実行する前に，プログラムを見てみましょう．

### LIST

左端に例の行番号がありますね．下から上へと，あわただしく表示されていきます．しかも，全てのプログラムが表示されるまで，止まりませんね．また，最初の方で表示された行は，いつの間にか，画面の上の方に消えてしまいました．今，画面に残っているのは，プログラムの終わりの数行だけです．では，もしプ

ログラムの始めの方を見たい時にはどうしたらよいでしょう．これには，画面を何とか止めておかないといけませんね．

やってみましょう．キーボードの左端にCTRLと書かれたキーがあります．見つかりましたか．このキーを，コントロールキーと呼びます．このコントロールキーをいつでも押せる準備をして，

```
LIST
```

表示され始めたら，CTRL-Sを入力します．画面は止まりましたか．止まらなければ，もう一度押してみます．今度は，止まりましたか．では次に表示を続けさせるには，どれかいずれかのキーを押します．試しにもう一度 CTRL-S を押します．どうです．動き出したでしょう．さらに，もう一度押すと止まります．しかし，この表示が止まった状態の時には，PC - 8801 は Ok のメッセージを返してきませんね．この状態では，プログラムを見るだけであればよいのですが，プログラムを修正したりはできません．そこで PC - 8801 から Ok のメッセージを返してもらうために，STOP と書かれたキーがあります．このキーは，STOP キーと呼ばれます．現在，表示が止まっている状態で，PC - 8801 から Ok のメッセージが出ていなければ，この STOP キーを押してみます．さあ，どうなりましたか．Ok が返ってきたでしょう．このようにして，あなたの見たいプログラムの場所を見ることができます．しかし，LIST とタイプすると，必ずプログラムの最初から表示されていくので，もしあなたが，プログラムの最後の方を見たい場合には，その場所が表示されるまで，じっと待っていなければなりませんね．プログラムが 500 行位あって，その中であなたが見たいのが，490 行目だとしたら，LIST とタイプしてから，489 行分を見送ることになります．では，こちらが行番号で指定した行を，自由に見ることはできないのでしょうか．いや，これができるんです．試しに，

```
LIST 100
```

どうです．次の行に100行が表示されましたね．では次に，

```
LIST 101
```

今度は，どうでしょう．何も表示されません．Ok だけですね．これは，このプログラムには101行という行番号は存在しないので，表示されなかったのです．では，100行台の行番号を持った行を見るには，どうしたらよいでしょうか．先程の LIST としてから表示を止めて調べますか．よい方法があります．

LIST 100-199

100と199の間に，マイナスの記号を使います．これで100行から，199行までに存在するすべての行が表示されましたね．こうしてLISTの後ろに行番号を指定することで，あなたの見たい場所を即座に表示させることが可能なわけです．

## 「走れ！」

いよいよ，プログラムの実行にかかります．御存知のように，RUN命令を使います．つまり，PC－8801に対して，「走れ！」と命令するのです．PC－8801は，RUN命令があなたからタイプされると，「それきた！」とばかりに，今まで覚えていた行番号と命令の中の，一番小さな行番号に書かれている命令から，順番に実行を始めます．やってみましょう．

また，LIST命令と同じように，RUN命令の後ろに，行番号を1つだけ指定できます．つまり，このプログラムの場合，100行から始まっているので，先程のRUN命令は，RUN　100と同じことになります．このように自分の好きな行番号から実行を始めることができます．しかし，当然その場合，その行番号から始めてもよいようにプログラミングされていることは言うまでもありません．また，存在しない行番号から始めようとしてもPC－8801を困らせるだけですし，何行から何行まで実行してもらおうと，RUN　100，199などとは，できません．RUNの後ろには，実行を始める行番号を一つだけ書くことが許されているだけです．

それでは，デモンストレーションを，当分お楽しみください．そして，目に焼きついたところで，次へいきましょう．

## そして何もなくなった

デモンストレーションのプログラムはいかがでしたか，そろそろ，目に焼きついたのではありませんか．それでは，NEW命令を使ってこのプログラムをすべて消すことにします．前にも登場したこのNEW命令は，現在PC－8801の中に入っているプログラムを一瞬のうちに消してしまいますから，NEWとタイプして↵（リターンキー）を押す前に，もう一度本当に今PC－8801に入っているプログラムは，消してしまってもよいのかよく考えましょう．まさに，後悔先にたたずのたとえそのものです．では，消したくない場合には，どうしたらよいのでしょう．これは，もうおわかりのように，カセットテープ等にそのプログラムを記録しておけばよいわけです．その方法については，後から，詳しく説明することにします．話を元に戻して，実際にNEW命令を実行してみましょう．その前に，

```
PRINT FRE(0)
 12228
```

次の行に数字がプリントされてきましたね．この数字の意味は，あなたが後どれ位のメモリを使うことができるのかを示しています．ちょっと難しいですね．なぜ，こんな数字をわざわざ出してきたのかといいますと，NEW 命令を実行することで，今まで，プログラムを覚えていた部分のメモリを取り戻すことができるということを示すためです．ではさっそく，

```
NEW
```

一瞬にしてデモンストレーションのプログラムは消えました．

```
LIST
```

確かにありませんね．

```
PRINT FRE(0)
 13308
```

次の行にプリントされた数字は，NEW 命令を実行する前にプリントされた数字よりも，大きくなっています．つまりこの二つの数字の差が計算できますね．そこに，デモンストレーションのプログラムが入っていたことになります．従って，大きなプログラムは当然たくさんのメモリを使うことになります．ですから，あなたが，大きなプログラムを使った場合には，一応この数字も気に留めておく必要があります．何故かというと，以前変数もメモリに記憶されると書いたように，同じメモリ上にプログラムとそのプログラムで使われている全ての変数の領域がとられます．正確には，まずメモリの最初からプログラムが入り，その後ろに変数の領域がとられていきます．従って，プログラムの領域と，そのプログラムで使っている変数の領域を足した大きさが，あなたの使えるメモリの範囲内に収まっていないといけません．しかし，まだ当分は，それほど大きなプログラムを作ることはありませんから，大丈夫です．

## さあ，もう一度復習しましょう

最後に，プログラムの入力，訂正，追加，削除についておさらいしておきます．

### プログラムの入力

行番号を付ければプログラムとして入力できることはわかりましたね．後は，いかにスクリーンエディットを駆使して簡単にプログラムを入力するかにつきま

す．しかし，慣れないキーボードからの，この入力の作業は，初めのうちは，地獄だと思います．めげずにがんばりましょう．

### プログラムの訂正

これもわかりますね．スクリーンエディットを使うことです．それも効率よくですよ．打ち直しは最後の手段と心得ましょう．

### プログラムの追加

行番号を使って行と行の間に，新しい行を挿入することは，できますね．ただそれだけです．しかし，追加を何度もしていくと，行番号がバラバラになってきますね．そこでその行番号をきれいな行番号に直すには，便利な命令があります．RENUN 命令といいます．使い方としては，実際にやってみましょう．

```
10 PRINT "LINE 10"
13 PRINT "LINE 13"
25 PRINT "LINE 25"
30 PRINT "LINE 30"
```

上のプログラムを入力します．

```
RENUM 10,10,10
LIST
10 PRINT "LINE 10"
20 PRINT "LINE 13"
30 PRINT "LINE 25"
40 PRINT "LINE 30"
```

行番号がそろいましたね．

```
RENUM 1,10,1
LIST
1 PRINT "LINE 10"
2 PRINT "LINE 13"
3 PRINT "LINE 25"
4 PRINT "LINE 30"
```

今度は，行番号が詰ってしまいました．

```
RENUM 1000,1,150
LIST
1000 PRINT "LINE 10"
1150 PRINT "LINE 13"
1300 PRINT "LINE 25"
1450 PRINT "LINE 30"
```

行番号が大きくなりました．どうです，おもしろいように行番号を付けかえることができますね．それでは，RENUM 命令に続く三つのパラメータの説明をします．だいたい見当はつくのではないでしょうか．

まず，最初のパラメータは，新しく付けかえる最初の行番号ということになります．ですから，新しく100行から始めたければ，ここに100を指定し，10行から始めたければ10を指定します。次に，二番目のパラメータは，現在付いている行番号の中のどの行番号から始めるかを指定します。ですから，ここの指定をかえることで，現在あるプログラムの一部分だけの行番号をつけかえることも可能な訳です．しかし，一部分だけ付けかえるとすると，注意しなければいけないことがあります．それは，新しく付けかえる行番号が，すでに存在する行番号と重なってしまってはいけません．

```
RENUM 1000,1150,150
```

どうなりましたか．エラーメッセージが返ってきましたね。その理由は，現在1000行から，プログラムがあります．その1000行はそのままにしておき，次の行である1150行を新たに1000行から行番号を付けかえようとしたので，新しくできる1000行と，すでに存在する1000行とがケンカしそうになり，コンピュータは，慌ててエラーメッセージを返してきたのです．最後の三番目のパラメータはもうおわかりですね．そうです．行番号の間隔を指定するパラメータです．この３番目のパラメータは書かなくてもよいのですが．

```
RENUM 10,1000
LIST
10 PRINT "LINE 10"
20 PRINT "LINE 13"
30 PRINT "LINE 25"
40 PRINT "LINE 30"
```

省略した場合，10を書いたのと同じに見なされます．そうなっているでしょう．これで，もうあなたは，いくらプログラムを追加していっても，RUNUM 命令によって，新たに行番号をきれいに付けかえることができるようになった訳です．

### プログラムの削除

削除したい行がある時には，行番号だけタイプして，↵（リターンキー）を押すことでできましたね．しかし，何行も続けて削除する場合，いちいち一行ずつ削除するのは，非常に不便です．そこで便利な命令があります．DELETE 命令です．

```
DELETE 20-40
LIST
10 PRINT "LINE 10"
```

10行だけが残りました．一度に，三行分削除できた訳です．もちろん，一行だけの削除もできます．

```
DELETE 10
LIST
```

全てのプログラムがなくなりました．DELETE命令の書き方は，LIST命令の書き方と同じです．ただその該当する行を画面に表示するか，削除するかの違いです．

これで一応プログラムを実際にどのように入力し，間違えた所はどのように修正をしたらよいのかは，自由にできるようになりました．また，プログラムの中の自分の見たい部分を見ることもできるようになりました．もちろん，プログラムの実行もできますね．でも，一つだけ，まだあなたにはできないことがあります．それは，実際にプログラムを作ることです．それには，もう少し新しい命令を覚えなければなりません．しかし，心配することはありません．順序を追ってやっていけば，それほど，難しいことではありませんから．一歩一歩着実にやっていきましょう．それが何よりも早道なのです．

## Basic PROGRAMMING

いよいよ簡単なプログラムを作っていく準備が整いました．すでにいくつかの命令も覚えました．あなたは，自信を持って読み進んでいって大丈夫です．また，プログラミングは難しいものだという観念も，同時に捨ててしまいましょう．どんなに大きなプログラムであっても，所詮は命令が一つ一つ集まったものですから，恐れることはありません．しかし，本書ではすべての命令を扱うわけではありませんので，もし自分の知らない命令が出てきた時には，よい機会だと思って，リファレンスマニュアル等を使って，その命令の使い方を調べ，その時点で自分の物にしてしまいましょう．知らず知らずのうちに，一度も見たことのない命令は姿を消していくことでしょう．そうなれば，もう恐いものなしです．後は，プログラムの流れを追うことに専念できるようになります．がんばりましょう．

## プログラミングにおける主な登場人物

これから新しい命令を五つあなたに紹介します．

```
INPUT
GOTO
IF....THEN....ELSE
STOP
END
```

さらに，

```
PRINT
```

は，すでにやりました．この合計6つの命令だけでも簡単なプログラムを作ることができます．各命令の説明をする前に，ひとつ作ってみましょう．

今までPC-8801に入っていたプログラムを全て消しておきます．

```
NEW

10 PRINT "<<  0 カラ N マデ ノ ソウワ  >>"
20 INPUT "N ヲ ニュウリョクシテクダサイ。";N
30 SUM=0
40 I=0
50 IF I>N THEN GOTO 90
60 SUM=SUM+I
70 I=I+1
80 GOTO 50
90 PRINT "0+...+N=";SUM
100 END
```

間違えずに入力できましたか．初めてプログラムの中にカナ文字が出てきましたが，英字と同じように入力してください．画面を一度クリアして，

```
LIST
```

よく見比べてみます．大丈夫ですか，されば，

```
RUN
<<  0 カラ N マデ ノ ソウワ  >>
N ヲ ニュウリョクシテクダサイ。?
```

Nを入力するように要求していますか．もし，エラーメッセージを返してきたのならば，プログラムの入力ミスです．もう一度，慌てずに見直しましょう．0から10までの総和が，55になるというのは御存知ですね．さっそく10を入力してみます．

```
N ヲ ニュウリョクシテクダサイ。? 10
0+...+N= 55
```

↵（リターンキー）も押すことも忘れずに．結果が出ました．確かに55になっ

ています．では100までの総和は，いくつになるでしょう．

```
RUN
<<  0 カラ N マデ ノ ソウワ  >>
N ヲ ニュウリョクシテクダサイ。?
N ヲ ニュウリョクシテクダサイ。? 100
0+...+N= 5050
```

少し時間をおいて5050がプリントされました．さて，ここで，数学の得意な人ならば，総和を求める公式を頭に浮かべていることでしょう．ちなみに0からNまでの総和は，

$$\frac{N(N+1)}{2}$$

で求められますね．しかし，一般にコンピュータで総和を計算させる場合には，このような公式を使うことは，あまりありません．今回のプログラムでも，同様に使っていませんね．確かにダイレクトモードで，

```
N=100
PRINT (N*(N+1))/2
```

一瞬にして5050が得られます．プログラムで実行すると，少し時間がかかりました．時間のかかる原因は，0から100まで正直に足しているからです．まさに，0＋1＋2＋3＋……＋Nの計算をしているのです．多少時間がかかりますが，人間の比ではありませんね．このように，わざわざ公式等を持ち出さなくても，短時間に計算してくれます．それでも，公式を使った方がよいのではないかという人に対しては，次の質問をします．0からNまでの偶数の総和を求める公式はどんな式でしたっけ．すぐに思い出せますか．それでは，奇数の総和は．3の倍数だけの総和は．何を言いたいのかわかりますか．わざわざ公式を使わなくても，この程度の単純な計算は，コンピュータの計算の速さに頼ってしまうのです．たとえば，先程の偶数の総和を求めるのであれば，プログラムでは，一つの数字を変更するだけで，容易に求められます．それは，70行のI＝I＋1の1を2に変えます．スクリーンエディットですぐにできますね。カーソルを70行の1の所に動かして，2をタイプしてから↵（リターンキー）を押します．さっそく，画面をクリアしてから，

```
RUN
<<  0 カラ N マデ ノ ソウワ  >>
N ヲ ニュウリョクシテクダサイ。? 10
0+...+N= 30
```

0＋2＋4＋6＋8＋10＝30．求まりました．これでもまだ公式を使おうと

思う人はいますか．やはり，単純な計算については，コンピュータの計算の早さにまかせましょう．

では，各命令の説明をしていきましょう．

### PRINT 命令

もう顔なじみの命令ですが，90行での使い方をみて下さい．〝で囲まれた後ろに；(セミコロン)で区切ってSUMという変数名がきています．このセミコロンを使うと，〝で囲んだプリントしたい文字のすぐ後ろに，変数(この場合はSUM)に入っている数値をプリントしてくれます．以前，ダイレクトモードでいくつかの数値を一つのPRINT文でプリントする場合，〝,〞(カンマ)で区切ったことを覚えていますか．PRINT文中での区切り記号としては，〝;〞と〝,〞が使えます．では，この区切り記号を入れないとどうなるでしょう．論より証拠，やってみます．90行の；を一文字削除します．得意のカーソルエディットです．リストが画面に残っていないのであれば，

```
LIST 90
90 PRINT "0+...+N=";SUM
```

そうそう，カーソルをSに持っていって，そうです．INSとDELと書かれたキーです．SHIFTキーは押さなくてよいですね．ポン．；が消えさりましたか．そうしたら，┛（リターンキー）です．

```
RUN
<<  0 カラ N マデ ノ ソウワ  >>
N ヲ ニュウリョクシテクダサイ。? 10
0+...+N= 30
```

あれ，さっきの表示の仕方と何も変わっていません．「なんだ，区切り記号なんて必要ないのか．」そうなんです．ある意味では，その通りです．しかし，この場合，〝と変数名との間の区切り記号だからこそ取ってしまっても；と同じ意味になりますが，考えてみてください．これが変数名と変数名との間の区切り記号だとしたら，どうなるでしょう．スペースは区切り記号として使えませんから，PC-8801は，どこまでが変数名なのか，わからなくなります．区切り記号を入れずに，いくつかの変数名を並べてプリントさせると，PC-8801は，ひとつの変数名だと勘違いします．

```
PRINT SUM;SUMSUM
 30  0
```

数値が二つしかプリントされません．つまりPC-8801はSUMという変数と，

SUMSUM という変数に入っている値をプリントします．結局，うまくこの区切り記号を使わないと思わぬ結果がプリントされることがあります．ご注意を！

### INPUT 命令

20行を見てください．「PRINT が INPUT になっただけではないか．」ちょっと待ってください．確かに，INPUT 命令の書き方は PRINT 命令と似た書き方をします．しかし，INPUT 命令というからには，何かを入力することができるはずです．このプログラムでも，N の値の入力をこの20行が行っています．20行をよく見てください．INPUT 命令は，確かに PRINT 命令と同じ働きも同時に持っています．つまり，INPUT 命令の後ろに〝で囲んだ文字をプリントしてくれます．しかし，ここからは PRINT 命令と大きく違います．〝で囲んだ文字の後ろに区切り記号の；があって，その後ろに N がありますね．この N の値は，今までプリントされましたか．されていませんね．つまり，INPUT 命令の後ろに変数名を書くと，その変数に好きな数を入力できるのです．気を付けてください．数ですよ．数だけですよ．今まで，プログラムを実行した時に，N の値を何度か入力してきましたが，これはプログラムとのやりとりをしていることなのです．もし，この入力命令がないとしたら，プログラムを実行して際中には，一切プログラムあるいはコンピュータとのやりとりができなくなります．またプログラムの中で，あらかじめ N の値を決めておかなければならなくなります．N の値を変えるには，プログラムの中で N に代入する値を，スクリーンエディットを駆使して変えなければなりません．ぞっとしませんか．入力命令には，INPUT 命令の他にも，いくつかあります．でも当分は，INPUT 命令だけで大丈夫です．

さて，それでは次に区切り記号を考えてみます．まずは，先程 PRINT 命令でやったように，；をとってしまいます．カーソルを動かして一文字削除です．

```
20 INPUT "N ヲ ﾆｭｳﾘｮｸｼﾃｸﾀﾞｻｲ｡"N
RUN
<<  0 ｶﾗ N ﾏﾃﾞ ﾉ ｿｳﾜ  >>
?SN Error in 20
```

ありゃりゃ．エラーメッセージです．INPUT 命令は，区切り記号を省略できないようです．では次に，一文字挿入．先程の；の位置に，（カンマ）を入れます．今度は，SHIFT キーも同時に押す必要がありますね．うまくできましたか．

```
20 INPUT "N ヲ ﾆｭｳﾘｮｸｼﾃｸﾀﾞｻｲ｡",N
RUN
<<  0 ｶﾗ N ﾏﾃﾞ ﾉ ｿｳﾜ  >>
N ヲ ﾆｭｳﾘｮｸｼﾃｸﾀﾞｻｲ｡
```

気が付きましたか．先程まで，N ヲ　ニュウリョク　シテクダサイ．という後ろに？が出ていたのですが，今回はどうですか．？が出てきません．信じられない人はとにかく，何か数字を入力してプログラムを終了させます．Ok が返ってきましたか．そして，先程一文字挿入した，（カンマ）を，；（セミコロン）にかえて，再びトライ！

```
20 INPUT "N ヲ ニュウリョクシテクﾀﾞｻｲ。";N
RUN
<<  0 カラ N マﾃﾞ ノ ソウワ  >>
N ヲ ニュウリョクシテクﾀﾞｻｲ。?
```

どうです．？が出てくるでしょう．つまり，INPUT 命令に使う区切り記号の；（セミコロン）と，（カンマ）の違いは，？を出すか，出さないかのちがいだけです．たいして気にする必要はありません．，（カンマ）の上に・（点）をつけると；（セミコロン）になるように，この点の部分で，？を出すか出さないかを区別して覚えておきましょう．，（カンマ）で区切ると？を出しませんよね．

INPUT 命令の最後として，先程，INPUT 命令は，数字だけの入力に使うとくどくどと書きました．数字以外を入力するとどうなるでしょうか？　英字のNを入力してみましょう．

```
N ヲ ニュウリョクシテクﾀﾞｻｲ。? N
?Redo from start
```

何だこれは，どうやら，もう一度入力するように言っているようです．

```
N ヲ ニュウリョクシテクﾀﾞｻｲ。? SUM
?Redo from start
```

やはり，だめですね．では，

```
N ヲ ニュウリョクシテクﾀﾞｻｲ。? 10
0+...+N= 55
```

うまくいきました．数字でないとだめなようです．数字以外の入力には，違う命令が用意されています．しかし，当分はおあずけ．まだまだ，それより先にやっておくことがあるのです．

### GOTO 命令

80行にある GOTO　50というのを見てください．1つ，あなたに謝まらなければならないことがあります．以前，RUN 命令の所で，プログラムの実行は，行番行の小さい行から順に実行されていくと書きましたが，かなずしも順に実行するわけではありません．この GOTO 命令を使うと，どこの行へでも，行番号を頼り

にして飛んでいくことができます．空を飛ぶわけではないのですが，一般にGOTO命令を使って何行へ飛ぶというように表現します．従って，GOTO命令の後ろには，行番号がくることはわかります．もし，その行が存在しないと，あなたは，エラーメッセージのお世話になることでしょう．このプログラムの場合，80行までくると，強制的に50行へ飛び，50行を実行します．ですから必ずしも，行番号の小さい行から大きな行へ実行をしていくとは限りませんね．このプログラムでも，80行から50行へ，大きい行から小さい行へ飛んでいる訳です．しかし，通常GOTO命令等が出てこない時には，最初にお話ししたように，行番号の小さい行から実行していくことに変わりはありません．プログラムを読んでいく場合，このことには十分気をつけてください．

### IF～THEN～ELSE命令

50行を見てください．えっ，ELSEがないって．そうです．この例はあまりよい例ではありません．この命令の使い方としては，プログラムを作っていく上で，必ず条件判断をさせたい場合が出てきます．つまり，もしこういう状態ならばAということを実行し，そうでないならばBということを実行する．このAとBとは命令と考えてください．そんな時には，このIF命令を使います．そしてIFの後ろには，条件を書きます．50行を見てください．IFの後ろに，I>Nと書かれていますね．意味としては，変数Iの値が変数Nの値よりも大きいとなります．ですから，I>Nという状態になっている時，条件を満足しているといい，それ以外，つまりI≦Nの状態（＝がはいることに注意）の時には，条件を満足していないといいます．そして，条件を満足した時には，THEN以下に書かれているELSEの直前までを実行し，ELSE以降は無視します．逆に，条件を満足していないのであれば，ELSEが表われるまで無視し，ELSE以降を実行しますが，もしELSEがなければ，どうなるでしょう．その時には，一般の実行順序に従い，実行します．この50行の場合も，もし変数Iの値が変数Nの値より大きいならば，THEN以下に書かれたELSEの直前まで（この場合にはＥＬＳＥはありませんが）の，GOTO 90を実行します．それ以外では，つまり，変数Iの値が変数Nの値よりも小さいか等しい時には，ELSE以降を実行しますが，何度も書くように50行にはELSEがありませんから，実際には，60行を実行することになります．おわかりですね，50行にELSEをつけることも可能です．

```
50 IF I>N THEN GOTO 90 ELSE GOTO 60
```

これでよい訳です．しかし，必要ありませんね．ちなみに，THENあるいは

ELSE の後ろに GOTO 命令を書く時には，この GOTO は書かなくてもかまいません．

```
50 IF I>N THEN 90 ELSE 60
```

これでも，構いません．ELSE 以下は必要ありませんから，

```
50 IF I>N THEN 90
```

これでよい訳です．さて，ここで：（コロン）の使い方について説明します．ダイレクトモードで，

```
PRINT "I=";I:PRINT "N=";N
I= 12
N= 10
```

次の行に I の値が，そしてその次の行に N の値がプリントされましたか．このように：（コロン）を使うと，一行の中にいくつも命令を書くことができます．これを，マルチステートメントと呼びます．先程の IF 文の中で使ってみると，（入力しなくてけっこうです．）

```
50 IF I>N THEN 90 ELSE SUM=SUM+I:I=I+1:GOTO 50
```

60行，70行，80行の三行分が，何と50行の ELSE 以降に収まってしまいます．しかし，このマルチステートメントを使うことで，欠点が一つでてきます．おわかりですか．それは，プログラムが見づらくなってしまうことです．先程の総和を求めるプログラムを，マルチステートメントを使って，書き直してみます（入力する必要はありません．）

```
10 PRINT "くく　0 カラ N マデ゛ ノ ソウワ　>>":INPUT "N ヲ ニュウリョクシテクダ゛サイ
。";N:SUM=0:I=0
20 IF I>N THEN PRINT "0+...+N=";SUM:END ELSE SUM=SUM+I:I=I+1
:GOTO 20
```

何と，二行になってしまいました．しかし，かなり見づらいプログラムになってしまいました．これは，少し極端な例ですので，無理もありませんが．しかし，マルチステートメントは欠点ばかりかというと，そうではありません．何故なら，メモリの節約になりますし，実行速度も，幾分大きくなります．従って，プログラムが見づらくならない程度に，マルチステートメントを使うのが良いでしょう．また，一行の中に，命令をいくつも書けるのですから．それでは，一体一行の中には最高でいくつかの命令が書けるのでしょうか．これに対して，最高で何個の

命令です，というような答え方は，残念ながらできません．というのは，一行の長さというものがすでに決められているからです．一行の中には，行番号を含めて255文字書くことができます．この255文字以内に収まっていればよい訳で，命令の個数には関係ありません．

今，あなたの見ている画面の桁数は，40桁ですか．それとも80桁でしょうか．40桁でしたら，255文字というのは，6行と半分位．80桁でしたら，3行とちょっとということになりますね．

### STOP 命令

ダイレクトモードで，

```
STOP
```

どうなりましたか．次の行に，

```
Break
Ok
```

続いて Ok が出ましたか．この STOP 命令は，その名前の通り，PC－8801 を止める時に，使います．ですから，Break というメッセージを，返してきたのです．

キーボードに，大夫慣れてきたあなたなら，キーボードの左隅に，STOP と書かれたキー(STOP キーと呼びます．)が目に入ってきたでしょう．押してみます．あれっ，今度は何のメッセージも返してきません．しかし，カーソルが一段下がりましたね．もう一度押してみます．やはり，カーソルが一段下がるだけですね．このように，STOP 命令と STOP キーとでは，使い方が少し違います．どのように違うのかといいますと，まず，STOP 命令は，プログラム中に書くことができます．当然ですね．しかし，プログラム中に書けるということは，逆に言うと，プログラムを実行した時に，この STOP 命令が書かれている行にきた時にだけ，プログラムの実行を止めます．ですから，あらかじめあなたが，この行にきたら実行をやめようと決めて，この STOP 命令をその行に書いておけば，プログラムは止まります．しかし，プログラムを止めたいと思うのは，そんな時ばかりとは限りません．つまり，プログラムを実行している際中に，急に，プログラムを止めたい時がしばしばあります．この時，当然，あらかじめここでプログラムを止めようとは考えていないはずですから，STOP 命令が入っていることを期待するのは無理でしょう．こういう場合には，STOP キーの登場となります．STOP キーを押すことで，PC－8801 は，今実行していた行の行番号を表示して，Ok を返してきます．たとえば，

```
Break in 40
```

などと表示されたなら，40行で実行を止めたことがわかります．そして，STOP命令あるいは，STOP キーでプログラムを止めた時には，たいていの場合，CONT命令を使うことで，再びプログラムの続きを実行することができます．これで，プログラムの続きを実行することができるようになります．プログラムを気軽にとめることができるようになりましたね．続けて実行したければ，

```
CONT
```

でOkです．何度か自分でやってみて，使い方に慣れましょう．これから先あなたが，もっともお世話になるキーは，このSTOP キーかもしれません．大切にしてあげましょう．

### END命令

総和を求めるプログラムの中に，初めてこのEND命令が出てきました．この命令は，プログラムの実行を終了させる命令です．しかし，実際には，このEND命令をプログラムの中に書かなくても，PC－8801はもう実行する行がなくなれば，実行を終了します．

## 監督はもちろんあなた

これで何とかあなたも，命令をいくつか組み合わせてプログラムができるようになったでしょう．まだ無理ですか．しかし，あなたは監督にでもなった気持ちで，覚えた命令をあなたの好きなように組み合わせて，あなたの考えていることを正確にプログラムの形にしてしまうことです．けれども，この作業が非常に大変だと気が付くのは，時間の問題でしょう．別に驚かすつもりはないのですが，とにかく大変な作業なのです．しかし，難かしく大変な作業だと思いながらも，次第に，このプログラムの世界に，どんどん引き込まれていく自分を発見することもまた，時間の問題となるでしょう．プログラミングの世界とは，そんな世界なのです．苦しいこともあるでしょうけれども，がんばりましょう．監督さん！

## 実行はコンピュータに任せる

監督のあなたが作ったプログラムは，PC－8801が実行してくれます．しかし，PC－8801は正直なもので，あなたの考えていた通りに実行してくれるとは限りません．PC－8801は，あなたがプログラミングしたプログラムを実行します．必

ず，プログラム通りに実行していきます．ですから，あなたが考えていた通りに，正確にプログラミングできた時に限り，PC－8801はあなたの考え通りに実行してくれます．それ以外の場合，つまり，あなたの考えが正確に，プログラミングされていない場合には，たいていあなたの意志に反して，PC－8801は実行します．こんな時には，プログラムの中に，あなたの考え方と異なった部分が必ずあります．このような部分を，バグ（虫）と呼びます．また，このバグを取る作業を，デバッグ（虫取り）と呼び，この作業がまたやっかいなものなのです．

## プログラムの保存は磁気の力で

あなたが，苦労して作ったプログラムでも，PC－8801の電源を切ったり，NEW命令を実行すると，一瞬にして消えてしまいます．どうがんばっても，なくなってしまいます．では，どうすればプログラムを保存しておくことができるのでしょう．もう，おわかりですね．カセットテープに記録することができる訳です．ミニフロッピーディスク装置をお持ちならば，ディスケットに記録しておくことができます．ここでは，カセットテープを使ってやっていくことにします．LISTを取ってみましょう．

```
LIST
```

先程の総和を求めるプログラムがまだ残っていますか．これを使ってみます．すでに電源を切ったり，NEWをしてしまった人は，短いプログラムを入力してください．どんなプログラムでも構いません．無理だという人は，デモンストレーション用のプログラムを，カセットから読めば良いでしょう．準備できましたか．尚，PC－8801とカセットレコーダとのつなぎ方等については，マニュアルを御覧下さい．

消してしまってもよいカセットテープをセットしたら，録音状態にします．リモート端子がつながっていれば，テープは回らないはずです．回ってしまった時には，一端，録音状態を解除して，

```
MOTOR 0
```

再び，録音状態にします．もうテープは回りませんね．もし回ってしまった時には，リモート端子がつながっていませんので，レコーダの方にある一時停止のボタンを押して、強引にテープを止めておきます．一時停止のボタンもないのであれば，録音状態を解除しておきます．

```
SAVE "CAS1:TEST"
```

↵（リターンキー）を押すのは，しばらく待って．ここで，録音状態をレコーダの方で止めている人は，一時停止を解除します．録音状態になっていない人は，録音状態にします．どちらの場合も，テープが回り始めましたね．リモート端子のつながっている人は，まだテープは回っていませんね．ここで↵（リターンキー）を押します．テープの回っていなかった人も，テープが回り始めたでしょう．これで，テープにプログラムが記録され始めました．後は，PC－8801から，Okが返って来るまで，そのままにしておきます．リモート端子がつながっていれば，Okが出ると同時に，テープは止まります．その他の場合には，テープは回り続けていますから，録音を解除します．次に，テープを巻き戻して，先程録音を始めた場所にテープをセットします．

```
LOAD? "CAS1:TEST"
```

↵（リターンキー）を押すと同時に，レコーダーを再生状態にします．後は，今までテープからプログラムを読んだ時と同じです．しかし，実際にはテープからプログラムを読んでメモリ（コンピュータ）に入れているのではなくて，テープに記録されているプログラムと，現在メモリに入っているプログラムとの比較をしています．つまり，メモリにあったプログラムを正確に，テープに記録できたかどうかを調べている訳です．この作業をベリファイと呼びます．Okが返ってくれば，正確に記録されたと考えて良いでしょう．少し面倒な作業ですが，プログラムをカセットテープに記録した時には，必ずこのベリファイをするようにした方が，後で泣かずに済みます．もし，Okが返ってこないで，エラーメッセージが返ってきたならば，音量と音質を調整してみて，再度ベリファイしてみますが，エラーメッセージが出た時には，もう一度，最初からやり直した方が（プログラムのセーブから）良いかもしれません．それはあなたにおまかせします．また，先程からテープにプログラムを記録するとか，カセットテープを読むとか書きましたが，これをそれぞれセーブする，ロードすると，呼びます．つまり，プログラムをカセットテープにセーブする，とか，カセットテープからプログラムをロードする，とか使います．従って，ロードするには，LOAD命令を使います．またセーブするのには，SAVE命令を使います．

```
LOAD? "CAS1:TEST"
Ok
```

ベリファイの結果，Okが返ってきたなら，一大決心をして，

```
NEW
Ok
LIST
Ok
LOAD "CAS1:TEST"
```

今セーブしたプログラムをロードしてみます．うまくいきましたか．

```
LIST
```

みごとに先程のプログラムがロードできました．カセットからいつでも，このプログラムをロードできるようになった訳です．

## カセットテープはこう使う

ロード，セーブ，ベリファイはすべてできるようになりましたか．簡単ですね．しかし，カセットテープにセーブすると，プログラムが長くなるに従って，かなり時間を必要とします．また，整理を怠ると，どこにどのプログラムがセーブされているのか，頭を悩ます結果となります．特に，両面で90分録音できるようなテープに録音していくと，片面でも45分の長さがあるので，目的のプログラムを探すのが，余計面倒になります．ですから，できれば短いテープに（たとえば両面で30分），きちんと整理してセーブしておくのが良いでしょう．でないと，後で本当に，大変な目に会います．これだけは保証します．気を付けましょう．

## ピーヒョロロ

プログラムをセーブしたカセットテープの音を聞いてみましょう．あなたには，どのように聞こえましたか．あまり音を大きくすると頭が痛くなります．この音の中には，実際には二種類の音しかありません．まさかとお思いでしょう．この二種類がいろいろと組み合わされて，たくさんの音に聞こえるのかもしれません．が，本当に二種類だけです．なぜ二種類なのでしょう．あなたは，いままでにコンピュータは二種数の世界で，０と１だけですべて行われているということを聞いたことはありませんか．そうなんです．この二種類は，実は０と１が区別されて録音されているのです．もう一度，聞いてみますか．どうです，０と１に聞こえてきましたか．聞こえてきませんか．

## 頭の痛くなるお話

少し，あなたの頭を悩ませましょう．何をするかといいますと，プログラムを一つ作ってもらいます．何のプログラムかというと，昔懐しい，２数の最大公約

数を求めるプログラムです．もう頭痛がひどくなりましたか．心配は入りません．順序を追ってやっていけば，必ずできます．そう信じて．

さて，最大公約数を突然求めようとしても，頭の中がどうにかなりそうですが，すぐにプログラムにはなりませんね．誰でもそうです．では，話を進めていきますので，頭痛がひどくなってきたら，一度休んでから，再び始めましょう．

まずは，あなたに質問します．整数M＝８，整数N＝４とした時の，MとNの最大公約数（G.C.M.）はいくつになるでしょう．やり方としてはいくつあるでしょうが，答えとして，４が得られましたか，M＝８＝$2^3$，N＝４＝$2^2$ ですから $2^2$ はどちらも，持っていますので，$2^2$＝4 としても良いでしょう．これをコンピュータにやらせてみます．気が遠くなりますか．そうですね．このやり方を，そのままやらせるのは，確かに大変そうです．では，コンピュータにやらせ易い方法を探しましょう．それには，ユークリッドの互除法というのがあります．一度位耳にしたことは，ありませんか．ユークリッドの互除法を用いて，先程のM，Nについて最大公約数（G.C.M.）を求めてみましょう．まずはMとNとを比べてみます．比較はIF命令で可能ですね．次に，Mの方が大きいかNと等しい時には，MとNはそのままにしておきます．逆にNの方が大きい時には，MとNの値を入れ替えます．つまり，M，Nの二数の内，大きい方の値をMに入れておく必要があります．また，値を入れ替えるというのは，こう考えて下さい．二本の牛乳をあなたの目の前に置きます．一本は白牛乳です．もう一本は，コーヒー牛乳です．そこで，この二本の牛乳の中身だけを，それぞれのびんに入れ替えたいとします．つまり，白牛乳の入っていたびんに，コーヒー牛乳を，コーヒー牛乳の入っていたびんの中に，白牛乳を入れたいとします．さあ，どうしたら良いでしょう．もちろん，まだ飲んではいけませんよ．方法としては，やはり別の容器を一つ用意しなければなりません．三本目の中身の入っていない牛乳びんを一本用意します．そして，順番はどうでも良いのですが，たとえば，白牛乳を空の牛乳びんに移します．次に，コーヒー牛乳の中身だけを，白牛乳の空いたびんに移します．そして，最後に，コーヒー牛乳のびんに，先ほど白牛乳を移したびんから移します．これでめでたく入れ替わりました．どうして，くどくどと，こんな話をしたのかというと，変数の値を入れ替えるのも，今と全く同じことです．新たに別の変数を一つ用意してから入れ替えないと，どちらかの変数の値を失うことになります．さて，ここまでをまとめてみます．変数M，Nに８と４を代入して次に，IF文を使ってM≧Nを満足したら，先に進みます．逆に，M≧Nを満足しなければ，MとNの値を入れ替えます．ここで，先程から，くどくどと話している新しい変数Sを用意します．そして，

```
S=M
M=N
N=S
```

これで，良い訳です．ここまでのプログラムは，もう頭の中に浮んできたことでしょう．さて次は，MをNで割った余りを求め，変数Lに代入します．M，Nには，実際には，8と4が代入されていますので，Lはいくつになりますか．そうです．L＝0となります．計算式は，どうなるでしょう．

```
L=M MOD N
```

と書くと，PC－8801は，MをNで割った余りをLに代入してくれます．また，

```
L=M-INT(M/N)*N
```

これでも，同じことになります．MODは，モードと読みます．INTは，インテェジャと読み，MをNで割った値の整数部だけを取ります．単純にMをNで割ってNをかけたのでは，元の値に戻ってしまいます．しかし，MをNで割った値の整数の部分だけにNをかけて，元のMの値から引いてやると，何とMをNで割った余りが求まります．実際にやってみると，よくわかります．

```
L=M-INT(M/N)*N
 =8-INT(8/4)*4
 =8-2*4
 =8-8
 =0
```

8を4で割った余りは，当然0になりますね．別の数でも，いろいろ試してみれば，すぐにわかるでしょう．さて，Lの値が求まりました．L＝0となったならば，Nの値を最大公約数（G.C.M.）としてプログラムの実行を終了します．この場合，L＝0になっていますから，最大公約数(G.C.M.)＝N＝4となります．先程の結果と同じになりました．では，L＝0になっていない時には，さらに先があります．今度は，変数の入れ替えではなくて，

```
M=N
N=L
```

として，さらにLの値を求めます．つまり，新たなM，Nの値に対してMをNで割った余りをLに代入します．後は，先程と同じことをL＝0になるまで，繰り返します．まとめてみます．繰り返しがありますね．この繰り返しのことを，ループと呼びます．コンピュータと，このループ（繰り返し）との関係は非常に密接で，このループをうまく使うことが，プログラミングのコツだと言っても良いの

です．ここで紹介するループは，あなたの目から見ると，ちょっとおかしくみえるかもしれませんが，気にしないで，先を読んでいかれることを期待します．あなたが，ちょっとおかしいと思える点は，必ず後から納得してもらえると思います．

では，そのループを形成してみます．まずは，Lの値，つまりMをNで割った余りを求めます．そして，次に，L＝0になっているか，比較します．もうわかりますね．IF命令ですよ．そして，条件（L＝0）を満足したら，Nの値をプリントしてプログラムを終了します．このIF文がループの頭になりますので，後からここへGOTO命令で飛んで来ますので，覚えておいて下さい．条件（L＝0）を満足しなかったのであれば，MにNを，NにLの値を代入して，Lの値を再度求めて，先程のIF文の頭に飛んでいきます．ここまでの説明で，すでにもうプログラミングを始めている人もいることでしょう．一応，プログラムのリストを載せておきます．

```
10 M=8
20 N=4
30 IF M<N THEN S=M:M=N:N=S
40 L=M-INT(M/N)*N
50 IF L=0 THEN 100
60 M=N
70 N=L
80 L=M-INT(M/N)*N
90 GOTO 50
100 PRINT "G.C.M.=";N
110 END
```

別に同じプログラムにならなくても，構いません．人それぞれが違うように，プログラムも違ってきて当然ですから……．

```
RUN
G.C.M.= 4
```

4という最大公約数（G.C.M.）が得られました．これが，ユークリッドの互除法をプログラミングしたプログラムです．しかし，プログラムと言うからには，もう少し汎用性が必要です．具体的には，M，Nの値は，やはり自由に入力したいものです．また，最後に最大公約数（G.C.M.）をプリントする時には，最初に入力したM，Nの値もプリントした方が良いでしょう．しかし，これ以降は，あなたにおまかせします．ヒントとしては，やはりINPUT命令を使いますが，MとNの値を最後まで残しておきたければ，一度別の変数に入力した後で，M，Nにそれぞれ代入することが必要です，何故なら，直接M，Nに値を入力してしまうと，プログラムを実行していく中で，値が変わってしまいますので，だめなの

です．従って，たとえば，MM，NN という変数に値を INPUT 命令で入力した後で

```
M=MM
N=NN
```

という具合に，M，Nに代入します．最大公約数（G.C.M.）をプリントするときには，この MM と NN の値を使えば，入力した値を知ることができますね．それでは，あなただけの，ユークリッドの互除法のプログラムを，がんばって作ってみてください．

## IF～GOTO～のループについて

```
IF 条件 THEN GOTO 行番号
IF 条件 THEN 行番号
IF 条件 GOTO 行番号
```

この三つの例のうち，間違った書き方をしてあるのは，何番目でしょう．三番目ですか．確かに，今までこの形をした IF 命令は出て来ていませんね．しかし，この三つの書き方は，どれも正しいのです．そして，上の三つのどの場合にも，後ろに ELSE 以降をつなげることもできます．一般に，ループ（繰り返し）をプログラム中に作る場合には，

```
IF 条件 GOTO 行番号
```

の形に IF 命令を使うと解り易いでしょう．もちろん，他の二種類の IF 命令でも構いません．好きな形の IF 命令をお使いください．それでは，簡単なループのプログラムを作ってみますので，現在 PC－8801 に入っているプログラムを消してください．必要ならば，消す前に，セーブしておくことを忘れないように．

準備できましたか．NEW 命令を実行してもらえましたね．では，新しいプログラムを入力していきます．

```
10 INPUT "ｶｲｽｳ=";K
20 IF K>255 GOTO 10
30 I=1
40 IF I>K GOTO 80
50 IF (I MOD 10)=0 THEN PRINT "*"; ELSE PRINT ".";
60 I=I+1
70 GOTO 40
80 END
```

打ち間違いがないと思ったら，

```
RUN
ｶｲｽｳ=?
```

回数を聞いてきますから，255以下の数を入力します．この回数というのは，ループを回る回数のことです．12と入力すると，

```
ｶｲｽｳ=? 12
.........*..
```

という具合に，10個おきに＊が出ますので，数える時の目安にしてください．プログラムの説明をします．このプログラムには，二つのループが含まれています．わかりますか．一つは，10行と20行で，もう一つは，40行から70行です．しかし，10行と20行とで形成するループは，回数として255より大きな数を入力した時に限ってループします．やってみましょう．

```
RUN
ｶｲｽｳ=? 300
ｶｲｽｳ=? 300
```

回数として，300を入力してみます．もう一度，聞き直してきましたね．もう一度，300を入力してみます．再度，聞き直してきました．では，－10を入力してみます．ループをマイナス10回実行してみようというものですが，そんな無茶な．御覧の通り，ループは一回も実行してくれませんね．マイナス10回とは，一体何回のことなのでしょう．では，０回はどうでしょう．

```
ｶｲｽｳ=? 0
Ok
```

０を入力してみます．確かにループを０回，つまり一回も実行してくれないことがわかります．このように，10行～20行，40行～70行のどちらもループを形成していることに変わりはありませんが，ループだからと言って必ずそのループを何回も実行するとは限らないことは，わかりましたね．しかし，10行～20行の形のループは，最低一回は実行します．でも，40行～70行の知のループは一回も実行しないことも有り得ます．０を入力した時には，確かに何もプリントされませんでした．つまり，ループの中にあるPRINT命令を実行しなかった訳ですね．このように，一般にループと呼ばれるものの中には，大きく分けて，二通りの形があることを覚えてください．これは，かなり重要なことです．ループを作ろうとする時には，まず，このことを考えてみて，必ず一回は実行するのか，それとも一回も実行してないことがあるのかどうか．それによってループの形を選んでください．とは言っても，二種のループの形が，まだよくわからないと思います．後からもう一度，これについて説明しています．ですから，わからなかった人は，

ループには，二種類の形のものがあるとだけ覚えておいてください．また，このプログラムはまだ活躍してもらうつもりですので，消さないで下さい．

## 恐怖の永久ループにあなたも襲われる

先程のプログラムの20行をカーソルエディットで，

```
20 IF K=K GOTO 10
```

と変更します．

```
RUN
カイスウ=?
```

回数を聞いてきました．あなたのお好きな数を入力してみてください．えっ，255以下の数を入力しても，また聞き返して来るって．マイナスの数でもだめですか．0でもだめですか．PC－8801が，壊れたのでないですか．いえ，心配はいりません．PC－8801は正常に，プログラム通り実行しています．プログラムが，間違っているのです．先程は，何気なく20行のIF命令の条件を変更してしまったでしょうが，よく条件を見てください．条件は，K＝Kとなっています．この場合の＝は，等号の意味で使います．代入文の時の＝とは区別して下さい．K＝Kとは，Kの値とKの値とが等しい時には，GOTO　10行へ戻ってしまいます．K＝Kとは，誰が考えても正しいですね．つまり，いつでも満足しているので，常に10行へ戻ります．このような状態を，永久ループと呼びます．いつまで立っても，ループを抜けることができない状態になってしまうことです．K＝Kなんて条件は，決して書くわけないって．そうです．確かにこんな条件を書くことはないでしょう．しかし，もう少し，条件が複雑になってくると，よくやってしまうのが，この永久ループです．今回の場合には，回数を入力するINPUT命令があるので，それほど不安ではないでしょう．プログラムの実行を止めるには，STOPキーでしたね．とにかくSTOPキーを押してプログラムの実行を止めます．もう一度20行を変更します．

```
20 IF K=K GOTO 20
```

条件を満足（実際はいつでも満足する）すると，20行つまり自分の行へ飛ぶ訳です．

```
RUN
```

回数として，適当な数を入力します．どうなりました．もう回数を聞いて来な

い替わりに，カーソルが，表示されませんね．ちょっと，不安になったでしょう．PC－8801は，20行だけを，いつまでも実行し続けます．STOPキーを押して，実行を止めさせましょう．永久ループとは，どんなものかわかりましたか．これから，あなたも自分の作ったプログラムの中にこの永久ループを作って，何度か自分自身を，襲うことになるだろうと思います．味方は，STOPキーとCTRL-Cだけです．他には何も……．

## ループ脱出計画

永久ループは，もう御免です．では，どうしたら永久ループを作らずに済むでしょう．そこで，このループ脱出計画を練ります．本当は計画書の提出をしてもらいたいものです．冗談はさておいて，永久ループを作らないコツは，やはり，条件の吟味をすることにつきます．それしかありません．がっかりしましたか．しかし，ループを作る以上，永久ループになる危険がありますので，いつループを脱出したらよいのかよく考えて，条件を設定します．先程のプログラムの20行も，元に戻してあげましょう．

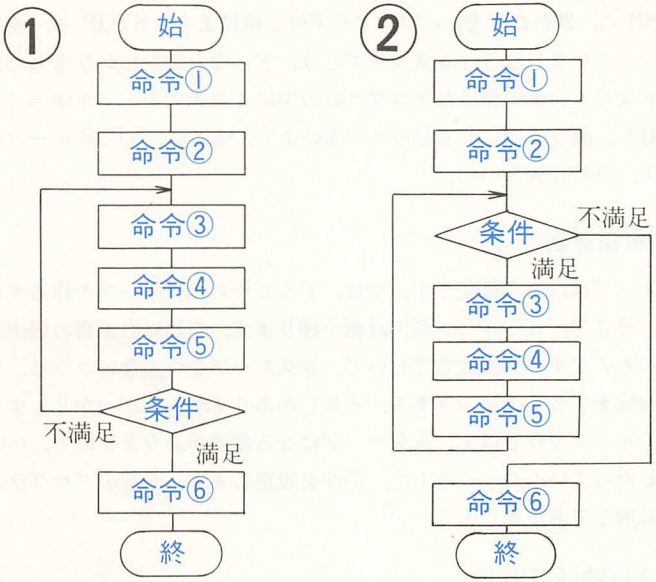
```
20 IF K>255 GOTO 10
```

これで，元に戻りました．

## もっとループを

ループを作る時の二種類の形，覚えていますか．ここで，さらに説明を加えます．以前の説明でよくわからなかった方は，じっくりと，わかった方も読み飛ばさずに，読んでみてください．新しい事柄も登場しますから．

さて，ループには，二種類の形があると前に書きました．覚えていますね．まず，最低一回はそのループを実行するものと，一回も実行しないことがあるものでした．しかし，よく考えて下さい．一回も実行しないループとは，最低一回は実行するループを含んでいることに気が付きましたか．だから，一回も実行しないことのあるループで，いつでもループを作っていれば良いような気がします．でも，これはうまいプログラミングであるとは言えません．やはり，この二種類を巧みに使い分けてプログラミングしていった方が良いでしょう．では，この二種類のループの形をはっきりさせるために，フローチャートと呼ばれるものを使って書いてみます．




①の方を見て下さい．四角の中に命令と書いてあるのは，GOTO 命令や IF 命令を除いた命令だと考えて下さい．ひし形の中に条件と書いてあるのが，そうです．あなたの得意な IF 命令です．ループしている場所は，わかりますね．命令③から命令⑤の間です．条件が不満足の時には，前の命令に戻ります．この場合，三つ手前の命令③に戻っていますね．これは別に三つ手前である必要はなく，手前の命令であればいくつでも構いません．ですから命令①でも命令②でも良い訳です．このループの形は，二種類のうちどちらでしょう．当然ループを最低一回は実行する形のループですよね．この形のループに名前を付けます．REPEAT～UNTIL 形ループと呼びます．条件を満足するまで，繰り返す訳です．それでは，②の方は残りの形のループということになります．御覧の通り，命令③から命令⑤は，一度も実行されないことがありえますね．つまり条件が，最初から不満足であれば，命令①・命令②・命令⑥を実行して終了します．条件が不満足であれば，命令③から命令⑤を条件が満足するまで繰り返します．こちらの形のループにも，名前を付けます．WHILE～DO 形ループと呼びます．条件を満足している間，ループを繰り返すわけです．この WHILE～DO 形ループを作る時には，一つだけ注意する点があります．何かというと，命令⑤の次には，GOTO 命令を使って飛ぶ訳ですが，この時には，先程の REPEAT～UNTIL 形の時のように，どこへ飛ん

でも良い，という訳にはいきません．飛ぶ場所は一箇所です．もちろん，IF 命令の所へ飛ぶことが許されるだけです．この WHILE～DO 形のループの場合，実際にプログラミングしていくと，たいてい命令①かあるいは，命令②の所へ飛びたくなってきます．何か思い当たることはありませんか．よ～く，胸に手を当てて考えてみてください．わかりましたか．以前に出てきました，ユークリッドの互除法のプログラムです．確か，割り算の余りを求める式，

```
L=M-INT(M/N)*N
```

が，二箇所に出てきていました．パラパラとページ P.OO)をめくって調べてみて下さい．確かに，40行と80行に全く同じ計算式が出ていますね．あの時のループは，どちらの形のループでしたか．もちろん，WHILE～DO 形のループでしたね．M が N の倍数になっている時など，

```
L=M-INT(M/N)*N
```

の計算をすると，L=0 となりますから，ループを一回も実行しないことがありえますので，WHILE～DO 形のループを使いました．そのため，同じ式が二箇所に出てくるはめになってしまったのです．何故なら，WHILE～DO 形ループの制約で，IF 命令に必ず飛んでループを作るために，IF 命令の一つ手前の命令には，飛べないからです．あなたからの反論が，聞こえてきました．「わざわざ同じ式を二箇所に書くのなら，IF 命令の一つ手前の命令に戻ってループを作った方がいいんじゃないか」と．その通りです．しかし，これだけは覚えておいて下さい．あなたの言われるように，ループの形を変更した時点で，もはや，WHILE～DO 形ループではなくなります．ただの名もないループになってしまいます．しかし，プログラムの長さが全体で一行減ることに，魅力を感じる方は，どうぞお使い下さい．

では何のために，この REPEAT～UNTIL 形と WHILE～DO 形のループを持ち出してきたのかと言いますと，一番の理由は，プログラムを見易くするためです．えっ，そんなに変りませんか．これが，あなたの目から見た場合にはそうですが，あなた以外の人が，あなたの書いたプログラムを見た場合には，たいへん違ってきます．あなたが，ループの形を二種類しか使わずに書いたプログラムを，この 2 種類の形のループを知っている人が見た場合，非常に解り易いプログラムということになります．そうですよね．WHILE～DO 形ループであれば，必ず IF 命令に飛んでくることは，見るまでもないのですから．また，ループの形が二種類であるということで，多少気が休まるかもしれません．逆に，あなたの好きな

ようにループを作ったプログラムの場合には，そういう訳にはいきませんね．この選択は，あなたにおまかせします．一つ書き忘れましたが，このREPEAT～UNTIL形と，WHILE～DO形ループの二つがあれば，どのようなループでも作ることができるということを，付け加えておきます．

## IF～GOTO～は万能選手

IF命令を使うと，REPEAT～UNTIL形でも，WHILE～DO形でも，その他の形のループのどれでも作ることができます．しかし，何度も書くように，条件を間違えると，偉いことになりますし，また，行番号を間違えても，そうです．使う場合にはくれぐれも，気をつけて使いましょう．それでは，REPEAT～UNTIL形とWHILE～DO形のループを使ったプログラムを載せておきます．

### REPEAT～UNTIL形ループの例

整数Nまでの階乗を求めてみます．どうしてREPEAT～UNTIL形のループを使うのか，わかりますか．ヒントは，0!＝1だからです．

```
10 I=0
20 S=1
30 INPUT "N=";N
40 I=I+1
50 S=S*I
60 IF I<N GOTO 40
70 PRINT "N!=";S
80 END
```

実行します．

```
RUN
N=?
```

入力するNの値は，0から33まで可能です．それより大きいとどうなるかって．あなたが，やってみて下さい．また，9より大きくなると，数字の表記法が変わりますので，0から9までが，入力する値としては無難でしょう．それにしても，階乗計算とは，すぐにバカでかい値になるものですね．ちなみに，エラーメッセージのお世話になった人はいませんか．

### WHILE～DO形ループの例

この形のループは，何度も登場していますので，書き方には慣れてきたでしょう．ここでは，あなたの入力するデータの総合計と平均を求めてみます．100行から入力して下さい．NEW命令は，実行する必要ありません．

```
100 S=0
110 INPUT "ﾃﾞｰﾀ ﾉ ｺｽｳ =";N
120 I=1
130 IF I>N THEN GOTO 190
140 PRINT "ﾃﾞｰﾀ(";I;")=";
150 INPUT D
160 S=S+D
170 I=I+1
180 GOTO 130
190 A=S/N
200 PRINT "(ｿｳｺﾞｳｳｸｲ) : ";S;"  (ﾍｲｷﾝ) : ";A
210 END
```

入力できたら，RUN ではありません．

```
RUN 100
```

です．最初にデータの個数を聞いてきますので，適当な数を入力しますが，当然その個数分だけデータを聞いてきますので，余り大きな数は避けましょう．せいぜい10個にしておきます．次にその個数分だけデータを入力します．テストの点数でも何でも構いません．入力が終わると，総合計と平均がプリントされて，プログラムは終了します．難しいプログラムではありませんね．最初の RUN 100 だけ注意して下さい．RUN としてしまうと，先程の REPEAT～UNTIL 形の例として入力した階乗を求めるプログラムが走ってしまいます．今回のプログラムは100行から入力してありますから，当然，

```
RUN 100
```

ということになります．

## FORとNEXTでサンドイッチ？

今まで，REPEAT～UNTIL，WHILE～DO 形のループは，全て，IF～GOTO 命令を使ってきましたが，ここで一つ新しい命令を紹介します．FOR～NEXT 命令と言います．ほら，すぐに顔をしかめないで！　この命令は非常に便利な命令なんですから．一部のループは，この FOR～NEXT 命令で書き替えることができるんです．実際にやってみましょう．先程の，階乗を求めるプログラムはまだ PC－8801 に残っていますね．他のプログラムは削除します．具体的には，

```
DELETE 100-210
```

で，OK です．不幸にも，階乗を求めるプログラムが残っていなければ，お手数ですが，もう一度入力しましょう．準備はできましたか．今度は120行から入力しま

すが，同じ部分がほとんどですから，スクリーンエディットを駆使しましょう．

```
120 S=1
130 INPUT "N=";N
140 FOR I=1 TO N
150 S=S*I
160 NEXT I
170 PRINT "N!=";S
180 END
```

入力が終わったら，画面を一度クリアして，

```
LIST
```

よく見比べて下さい．先程より一行短かくなりました．頂度，ループの前後を，FOR と NEXT でサンドウィッチする形になっていますね．これで，先程のプログラムと，全く同じことを実行します．RUN と RUN120で，調べてみてください．同じ結果が得られるでしょう．FOR～NEXT 命令を使うと，一般に，IF～GOTO 命令を使って書いたプログラムより，一行分少なくなります．しかし，FOR～NEXT 命令には，IF 命令と同じような条件を書くことはできません．このために，全てのループを，FOR～NEXT 命令だけで書くことはできません．というのは，140行を見て下さい．

```
FOR I=1 TO N
```

これは，変数Ｉの値が１からＮになるまで繰り返されます．もう少し正確に言うと，プログラムの実行が，140行までくると，変数Ｉに１が代入されます．次に TO 以下に書かれている値（この場合はＮ）と比較してその値と等しいか小さければ，NEXT が書かれている所まで，次々に実行します．NEXT が現われると，変数Ｉに１が足されます．それから，TO 以下に書かれている値（この場合はＮ）と比較してその値と等しいか小さい限り，FOR と NEXT が書かれている間にある命令を繰り返し実行します．そして，Ｉの値が TO 以下に書かれている値（この場合はＮ）を越えた時には，NEXT の次の命令を実行していきます．つまり，ループから脱出する訳です．従って，条件としては，いくつからいくつまでという指定しかできません．

```
FOR J=1 TO 10
NEXT J
```

と書けば，変数Ｊが１から10まで，つまりループを10回実行することになります．この変数Ｊに最初に代入される値（この場合は１）を，初期値と言い，TO 以下に書かれた（この場合は10）比較の対象となる値を終値と言います．先程は，NEXT

の所へくると変数Ｉに＋１されると書きましたが，この値も設定することができます．この値のことを，増分といいます．

```
FOR K=0 TO 4 STEP 2
NEXT K
```

と書くと，変数Ｋには，２が足されていきます．つまり，変数Ｋには，最初，初期値として１が代入されて，NEXT へくると＋２されて，３となります．３は５よりまだ小さいですから，もう一度ループが実行され，再び NEXT へくると，さらに＋２されて，５となります．５は，終値４を越えていますので，ループを脱出する訳です．従って，この場合ループは二回実行されることがわかります．STEP を書かないと，STEP　１と書いたのと同じに見なされます．もちろん、マイナスの値を増分とすることもできます．増分がマイナスとは，ちょっとおかしく感じますね．

まず最初に，

```
FOR L=N TO 1 STEP -1
NEXT L
```

これは，変数Ｌに初期値Ｎが代入されて，NEXT へ来る度に，変数Ｌが－１されていき，Ｌの値が，１より小さくなった時点で，ループを脱出します．

NEXT の後ろには，何が来るのかわかりますか．初期値が代入される変数名を，NEXT に続けて書いておきます．しかし，この変数名は省略して，単に NEXT だけでも構いません．大事なことは，作ろうとするループを FOR と NEXT ではさむことです．忘れないように．

## FORの中にまたFOR

プログラムを，入力します．

```
NEW
10 FOR I=1 TO 3
20 FOR J=1 TO 3
30 PRINT "I=";I;"   J=";J
40 NEXT J
50 NEXT I
60 END
```

実行する前に，ＩとＪの値がどのように変化するのか，考えてみます．このプログラムは，二重ループと呼ばれるもので，ループ中にさらにもう一つループがある訳です．わかりますか．つまり，10行から50行の間が，変数Ｉの外側のループになっていますね．そして，その中の，20行から40行の間が，変数Ｊの内側のループになっている訳です．このように，ループが幾重にも重なっている状態を，ネ

スティング(入れ子)の状態と呼び，よく使われる形です．変数Ｉと Ｊの値がどうなるか，想像が付きますか．実行してみましょう．

```
RUN
I=  1    J=  1
I=  1    J=  2
I=  1    J=  3
I=  2    J=  1
I=  2    J=  2
I=  2    J=  3
I=  3    J=  1
I=  3    J=  2
I=  3    J=  3
```

９行に渡って，変数Ｉと Ｊの値がプリントされましたね．プログラムとよく見比べて，どのように実行されたのか，調べてください．この外側のループと内側のループの概念は，わかりますか．それほど，難しくないでしょう．ただし，気を付けることがあります．たとえば，40行と50行をひっくり返したりすると，外側と内側のループの対応がうまくいかなくなりますね．ですから，ループを何重にもしていく場合，NEXT の次に書く変数名は，一番最近 FOR の次に書いた変数名から，だんだんと古い変数名へと逆の順序で書かなければなりません．また，単に40行と50行を一行にまとめて，

```
NEXT J,I
```

というように書いても，構いません．ちなみに，このプログラムと同じことをするプログラムを，IF 命令を使って書いておきます．いかに，FOR～NEXT 命令を使うと，簡単にループを作れるかということが，わかると思います．

```
10 I=0
20 I=I+1
30 J=0
40 J=J+1
50 PRINT "I=";I;"   J=";J
60 IF J<3 GOTO 40
70 IF I<3 GOTO 20
80 END
```

## 初歩のプログラミングヒント

プログラミングを始めてすぐに，あなたは，大きな壁にぶつかるだろうと思います．とにかく，自分の思い通りには，なかなか PC－8801 は実行してくれなくて，イライラすることでしょう．コンピュータとは，こんなに世話のやける代物なのかと思う位に．しかし，ここで，止めてしまったのでは，先へは進めなくなりま

す．そんな時には，これから書いていく，プログラムヒントを，思い出してください．

**エラーメッセージ続出のあなたへ**

エラーメッセージが返ってくるのは，まだ上等なことだと考えましょう．エラーメッセージが返ってこないプログラムに，バグがいる時のことを考えましょう．あなたよりもっと悲惨ですよ．誰でも，エラーメッセージを出して大きくなっていくものなのです．

**永久ループに悩むあなたへ**

もう一度，ループ脱出条件をよく調べます．条件が正しいのであれば，ループの中でその条件が，満足あるいは不満足の状態になることが決してありませんから，条件を設定している部分を調べます．必ず見つかりますよ．でも，たまには，STOP キーもいたわってあげましょう．

**バグがどうしてもみつからないあなたへ**

なかなか見つからないからこそ，バグとして価値があるのです．他人の意見を聞いてみるのも良いでしょう．時間をおいてから調べるのも良いと思います．手こずった末に，やっとバグが見つかった時の喜びは，口では言い表わせないものがあります．えっ，あなたはデバッグで，今日も徹夜ですか．御愁傷さま．

**その他のあなたへ**

何も言うことは，ありません．プログラミングの楽しさが，大分わかりかけてきたのではないでしょうか．後は，あなたの努力次第です．がんばりましょう．

## :(コロン)と;(セミコロン)，.(ピリオド)と,(コンマ)は同じようでも大違い

．(ピリオド) を除いて，残りの 3 つの使い方は，もうおわかりですね．：(コロン) は，マルチステートメントで，命令と命令を区切るために使いました．マルチステートメントは，ダイレクトモードでも，プログラムモードでもどちらでも使えます．；(セミコロン) は，PRINT 命令あるいは INPUT 命令の中で使いましたね．キーボードでは，：(コロン) と；(セミコロン) は，隣り同士です．間違えないように，気を付けましょう．，(カンマ) も，；(セミコロン) と同じく，PRINT 命令あるいは INPUT 命令の中で使いました．．(ピリオド) は何に使

えるでしょう．もちろん，小数点として使えます．またもう一つ別の使い方があります．現在，PC－8801の中にプログラムが入っている人は，試してみてください．STOPキーをいつでも押せる状態で，

```
LIST
```

を実行し，全てのプログラムを表示し終わる前に，STOPキーを押して，表示を途中でストップさせます．続いて，

```
LIST . -
```

どうです．先程の続きが，表示されるでしょう．．（ピリオド）は，こんな使い方もできるのです．．（ピリオド）と，（カンマ）も，キーボードでは，隣り同士にあります．間違えないように，気を付けましょう．

## まとめ

この章では，プログラミングの初歩について，いろいろ説明してきました．もちろん，ここで説明した命令の他にも，もっとたくさんの命令が使えます．リファレンス・マニュアルを見てもらえば，使える命令の全てを知ることができます．何を言いいたいのかというと，本書に出てこない命令について，あなたはその使い方を知らないからと言って，そのままにしておかないで，あなた自身が実際にリファレンス・マニュアルのページをめくって，その命令の使い方の説明を読み，PC－8801で実行してみてほしいのです．どうかリファレンス・マニュアルを，これからは，あなたの手許において，今度はリファレンス・マニュアルの方から，いろいろなことを吸収していってください．リファレンス・マニュアルには，あなたの知りたいことが一杯詰っているのですから．

# 第4章

# グラフィックスがいっぱい

## 第3章まで読んでこられた方へ

第3章まで，一応，ダイレクトモードとプログラムモードの二つのモードで，PC－8801を使えるようになりました．これから先，第4章と第5章では，これらの二つのモードを取り混ぜて使っていくことになります．特に，プログラムモードで使う場合，第3章で説明した命令以外の命令が，たくさん出てきます．これまで，何度も言っているように，そのような時には，迷わず，リファレンス・マニュアル等で，その命令の使い方を調べて下さい．また，初めて出てくる命令はリファレンス・マニュアルを参照してください．しかし，この第4章では，PC－8801のグラフィックス命令の説明をしていく訳ですから，グラフィックス命令に関しては詳しい説明をしていきますので安心して下さい．突然，グラフィックス命令などと，聞き慣れない言葉が出てきました．PC－8801は，このグラフィックス命令というものを持っています．

何をする命令かといいますと，第1章をちょっと思い出してもらうとわかると思います．第1章で，あなたは，2枚の同じ大きさの画面を重ねて見ているという事を説明しました．覚えていますか．そして，1枚目の画面には，キーボードから入力した文字を表示させる事ができると説明したはずです．そうでしたね．気が付いた方もいるかもしれませんが，この表現の仕方は，少しおかしいのです．キーボードから入力した文字が，すべて画面に，表示はされませんね．たとえば，TABと書かれたキーを押しても，画面にTABと表示はされません．ですから，言い直します．1枚目の画面には，特別な働きをするキーを除いて，キーボードから入力した文字を表示させる事ができます．

さて，それでは，もう一枚の方の画面には，何が表示できるのでしょうか．こちらの画面には，第1章ですでに説明したように，たくさんの点を打つ事ができる訳です．そして，グラフィックス命令とは，この二枚目の画面の好きな場所に，点を打ったり，打たなかったりすることのできる命令だということになります．点を打つという表現が，よくわかりませんか．これは，画面上の一点（これをドットと呼びます）を，点灯させたり，消灯させたりする事だと考えて下さい．このドットの個数は，第1章で触れたように，気の遠くなるような個数がある訳です．

これからの説明の都合上，先程の二枚の画面に名前を付けておきましょう．一枚目の画面の方，どちらの画面かわかりますか，キーボードから入力した文字が表示される方の画面を，テキスト画面と呼びます．残った二枚目の画面の方，つまり点を打ったり，打たなかったりできる方の画面を，グラフィックス画面と呼

びます．ですから，現在，あなたは，テキスト画面と，グラフィックス画面の二枚の同じ大きさの画面を重ねて見ている訳です．多分，テキスト画面には，何かしら文字が表示されているでしょうが，グラフィックス画面の方は，画面のすべてのドットが消えているはずです．ですから，テキスト画面を，一枚だけしか見ていないように思えるでしょう．無理もありません．けれども，しだいに，グラフィックス画面を使っていきますから，この二つの画面の区別が，はっきりする事でしょう．

## とにかく描いてみる （キャラクタ）

第1章で書いた事を，思い出して下さい．あの，グラフィックシンボルキーの所です．キーボードのGRAPHと書かれたキーを押しながら，別のキーを押すと，画面に，グラフィックシンボルが表示されましたね．

もちろん，すべてのキーにこのグラフィックシンボルがあるのではなくて，第1章の所で説明した時に，キーボードの図が一緒に出ています．合計56個のシンボルがありました．この56個を使って，テキスト画面に，ある程度のグラフィックスを表示させる事は可能です．

やってみましょうか．円を描いてみます．円といっても，漢字の円では，ありません．丸の事です．キーボードから入力して下さい．

```
WIDTH 40,25
```

GRAPHキーを押しながら，QとWの書かれたキーを一回ずつ押します．

次に，GRAPHキーを放して，カーソルを一段下げて，2つ左へ動かします．

また，GRAPHキーを押しながら，AとSの書かれたキーを一回ずつ押します．

```
O
```

どうです，円になったでしょう．テキスト画面の，4文分で，1つの円が描けました．もう一つ別の円を描いてみます．

カーソルを，右へ3つ動かします．GRAPHキーを押しながら，Hの書かれたキーを押してみます．

```
O   o
```

今度は，1文字分で，1つの円が描けました．もう少し，ちがった大きさの円を描きたいのですが，56個のグラフィックシンボルの中には，2種類の円しかありません．そこで，実際にキーボードに書かれている文字の中から，円に近い形を探してみましょう．

カーソルを，右へ3つ動かして，SHIFTキーを押しながら，Oと書かれたキー

を押します．大文字のOが表示されましたね．

さらに，カーソルを3つ右へ動かして，今度は，小文字のoです．SHIFTキーは押さなくてよいですね．

O　o　O　o　.　.

これで，合計4種類の円が表示されましたね．あと2種類，付け加えます．

カーソルを3つ右へ動かします．カナキーを押します．さらに，SHIFTキーを押しながら，ルと書かれたキーを押します．場所は，右側のSHIFTキーから，3つ左にありますね．

カナキーをもう一度押して，カナモードを解除してから，カーソルを3つ右へ動かしたら，先程と同じ，ルと書かれたキーを押します．SHIFTキーは押しません．

これで，合計6種類の円が表示されたでしょう．えっ，どうみても，円には見えないって．そうかもしれません．

これが，グラフィックシンボルを使った限界です．また，一つ注意してほしいのは，あくまで，テキスト画面に表示させている事です．まだ，グラフィックス画面には，表示させていませんね．それでは，グラフィックス画面を使うと，どんな円が書けるかって．それは，もう少し後からの，お楽しみです．とりあえず，ここでは，テキスト画面を使っていたのでは，どうやってみても，先程の6種類の円しか書けないということだけ，しっかりと頭に入れておいて下さい．

## まず最初にすることは　（モードの設定）

いよいよ，グラフィックス画面に描いてみましょう．その前に，どうも気を持たせるようですが，グラフィックス画面を使える状態にしなければなりません．それには，グラフィックス画面を使う上でのモードを3つあなたに，覚えてもらいます．

1)カラー　　(640×200ドット・1ページ)
2)白黒　　(640×200ドット・3ページ)
3)高分解能白黒　(640×400ドット・1ページ)

＊：但し3)のモードは専用の高分解能モニタを接続しなければなりません．

以上の3つのモードがあります．

1)は，カラー画面で，横に640個，縦に200個の，合計640×200＝128000個のドットをつけたり，消したりできます．

2)は，1)と同じドット数の画面を，3ページとれますが，カラー画面ではなく

て，8色の中のいずれか一色となります．3ページというのは，1)の画面と同じ大きさの640×200ドットの画面を，3枚分という訳です．そして，この3ページのどのページを表示させるかとか，どのページに書き込むかという選択ができる訳です．わかりますか．つまり，見ている画面以外の画面に書き込む事ができるのです．その場合，当然見ている画面には表示されません．別のページに書き込んだものを表示させるには，書き込んだページを表示させなければなりません．

3)は，2)と同じ様に，カラー画面ではなく白黒で，ページ数は1ページです．何がちがうのかというと，ドット数が，今までの倍になります．横方向は，同じ640個ですが，縦方向には，400個となり，合計640×400＝256000個となります．ちょっと，気が遠くなる個数ですね．

以上の3つのモードの切り変えには，SCREEN という命令の最初のパラメータを使います．

```
1)のモードは，SCREEN  0
2)のモードは，SCREEN  1
3)のモードは，SCREEN  2
```

という具合です．SCREEN には，合計4つのパラメータを付ける事ができますが，残りの3つのパラメータについては，リファレンス・マニュアルを調べて下さい．これから先，1)のモードを使っていきますので，

```
SCREEN 0
OK
```

を実行して下さい．いよいよ，準備が整いました．

## 画面をクリアする

この見出し，前にも出ましたね．なぜ，再び登場したのかというと，テキスト画面と，グラフィック画面とでは，別々にクリアする事ができるのです．つまり，テキスト画面だけをクリアしたり，グラフィックス画面だけをクリアしたり，両方の画面を一緒にクリアしたりできる訳です．今までは，テキスト画面だけしか使いませんでしたから，画面のクリアには，

```
CLS
OK
```

あるいは，

上段に HOME，下段に CLR と書かれたキーを押す．の，どちらかで，画面はクリアされました．しかし，これから，グラフィックス画面に何か描き込むと，

前の二つの方法では，決してクリアできません．では，どうすればよいのかというと，

```
CLS 2
Ok
```

と，CLS の後ろに，2 を付けます．これで，グラフィックス画面のクリアができます．ちょっとやってみましょう．

```
CLS 2
Ok
```

数秒後に，Ok と表示されましたね．この数秒の間に，グラフィックス画面をクリアしている訳です．現在，グラフィックス画面には，何も表示されていないはずですから，わかりにくいかもしれませんが，グラフィックス画面のクリアには，CLS 2 を使います．次に，テキスト画面も，同時にクリアしたい時には，

```
CLS 2:CLS
```

と，しますか．そんな，めんどうな事はいりません．

```
CLS 3
Ok
```

これで，大丈夫です．CLS 3 を実行すると，まず，テキスト画面がクリアされ，続いて，グラフィックス画面が，クリアされます．やってみれば，すぐにわかります．ここまで書くと，では，CLS 1 は，何をクリアするのか，疑問でしょう．CLS 1 は，CLS と同じなのです．つまり，CLS の後ろに，何も書かないで省略してしまうと，1 を書いたのと，同じに見なされる訳です．ですから，今まで，CLS と書いてきましたが，これは，CLS 1 と書いてきたのと同じだったのです．では，CLS 0 は，どうなるでしょう．これは，実際に，やってみればわかりますので，あえて書きません．以上，まとめて書いておきます．

テキスト画面をクリアするには，

CLS
CLS 1
HOME・CLR キーを押す

グラフィックス画面をクリアするには，

```
CLS 2
```

テキスト・グラフィックス画面を両方をクリアするには，

```
CLS 3
```

## 点を打つ

とうとう，グラフィックス画面に，点を打つ所までやってきました．しかし，ここで，最後のお願いがあります．これから，やっていく事が，誰がやっても同じになるように，次の事を実行しておいて下さい．

```
CONSOLE 0,25,0,1
Ok
COLOR 7,0,0,7
Ok
FOR I=0 TO 7:COLOR=(I,I):NEXT
Ok
```

何をしているのか，あまり深く悩む必要はありません．上の事を，確実に実行しておいて下さい．もちろん，エラーメッセージが返ってきたりしたら，どこかを打ち間違えていますから，直してから，再びやって下さい．すべて実行されると，ホームポジション(画面左上隅)に Ok と表示され，その次の行にカーソルが，チカチカとしているはずです．そうなっていますか．そうなっているのであれば，

```
PSET(20,4)
Ok
```

一行目に表示されている Ok の横を，よく目を疑らして見てみましょう．点が打たれていませんか．見えない方には，さらに，

```
PSET(21,4)
Ok
```

今度は，見えますね．これで，グラフィックス画面に点が打てました．テキスト画面ではない証拠に，HOME・CLR キーを押して，テキスト画面をクリアしてみます．どうです．まだ，カーソルの横に，先程の二点が残っているでしょう．この PSET 命令を使うと，好きな場所に，点を打つ事ができる訳です．しかし，せっかく，カラー画面のモードを使っているのですから，他の色も出したいですね．

```
FOR I=1 TO 7:PSET(20,I*8+4),I:PSET(21,I*
8+4),I:NEXT
Ok
```

7色出ましたか．よく見て下さい．上から，青，赤，紫，緑，水色，黄色，白の順番です．テキスト画面は，クリアしておいた方がよいでしょう．

```
CLS
Ok
```

を実行します．グラフィックス画面は，残りますね．

では，PSET 命令の使い方を説明します．まず，PSET に続けて，横方向と縦方向のドットの位置をカッコでくくって指定します．この時，画面の左上隅が，（0，0）となります．したがって，画面の右下隅は，(639，199）となりますね．横方向，縦方向どちらも，0から始まりますから，合計で，640×200＝128000個のドットの場所を選べることになりますね．それから，先程やったように，カッコに続けて，カンマで区切って，色の指定ができます．何も書かなければ，COLOR 命令の四番目に書いたパラメータが使われます．そこで，一番最初にやった設定の所を見て下さい．CONSOLE 命令に続けて，COLOR 命令を実行していますね．その COLOR 命令の四番目のパラメーターは，いくつですか．もちろん，7になっています．ですから，PSET を，初めに実行した時に，色の指定はしませんでしたが，実際には，7で色の指定をしたことと同じだった訳で，白で点が打たれたのです．では，COLOR 命令の四番目のパラメータとは，一体何を設定するのかという事になりますので，ここで，COLOR 命令の説明をしておきましょう．先程，

```
COLOR 7,0,0,7
```

と，実行しました．すでに，COLOR 命令の一番目のパラメータについては，第2章で触れました．覚えていますか．そうです，表示される文字の色を変える所でやりました．この一番目のパラメータを変える事で，合計8色の色が出せましたね．もちろん，画面が，カラーモードになっていなければなりません．ちなみに，カラーモードではなく，白黒モードの時に，このパラメータを変えてみましたか．まだの方は，一度やっておくとよいでしょう．その時，1，3，5，7を指定すると，文字が見えなくなりますので，気を付けて下さい．カラーモードの時に，COLOR 0 を指定したのと同じ状態になります．白黒モードでの，COLOR 0 は，通常の表示となります．少し，ややこしいですから詳しい事は，リファレンス・マニュアルの方をどうぞ．では，2番目のパラメータはというと，これについては，次で触れます．最後に，3番目のパラメータですが，これは，やってみれば，すぐにわかります．

```
COLOR 7,0,7,7
Ok
```

何が変わったか，わかりますね．画面にふち取りの色が白で付いたでしょう．この色の変わった部分については，あなたは，文字を書くことはできません．唯

一，COLOR 命令の三番目のパラメータで色を変えることができるだけです．色色（？），試してみて下さい．合計 8 色の色が付けられるのは，御存じの通りです．さて，4 番目のパラメータにいきます．先程から問題の，4 番目のパラメータは，フォアグラウンドカラーの指定をします．フォアグラウンドカラーなどと，聞き慣れない言葉が出てきましたが，これは，PSET 命令等のグラフィック命令に対して，色の指定を省略した時には，このパラメータの値が使われることになります．実際にやってみると，

```
COLOR 7,0,0,2
Ok
PSET(20,4):PSET(21,4)
```

どうです．今まで，白で点を打たれていた場所が，赤に変わったでしょう．2 は赤色と対応していましたね．それで，色を指定しなかったので，COLOR 命令の 4 番目のパラメータである 2 が，色として使われた訳です．

次に進む時には，

```
COLOR 7,0,0,7
Ok
```

を実行してから，進みましょう．

## 点を消す

やっと，グラフィックス画面上にドットを表示することができるようになりましたが，それでは，現在あるドットを消すには，どうしたらよいでしょうか．しかし，よく考えてみて下さい．ドットを黒で打つというのは，どういうことでしょう．やってみましょう．

```
PSET(30,4)
Ok
```

どこに，ドットが打たれたかわかりましたか．次に，

```
PSET(30,4),0
Ok
```

どうです．ドットが消えたでしょう．という訳で，表示されているドットを消す時には，そのドットを今度は，黒で打ってやれば，見事に消える事になります．ドットを表示させるのも，消すのも，共に PSET 命令を使えばよいようです．でも，次の場合はどうでしょう．

```
COLOR 7,2,0,7
Ok
CLS 2
```

画面の色が変わりましたね．四角い赤色の部分が現われました．同じように，

```
PSET(30,4)
Ok
```

表示されたドットを確認した上で，さらに，

```
PSET(30,4),0
Ok
```

同じように，黒で先程のドットの上から表示すると，何と，当然の事ですが，黒でドットが残ってしまいました．したがって，ドットの表示を消す時は，常に黒で，ドットを表示させればよいという訳にはいきませんね．そこで，どうしたらよいのかというと，便利な命令が用意されているのです．PRESET 命令といいます．

```
PRESET(30,4)
Ok
```

どうです，先程のドットが，なくなったでしょう．別に，わざわざ PRESET 命令を使わなくても，

```
PSET(30,4),2
```

と，やればよいのですが，色指定に気を付けなければなりません．やはり，ドットを消す時には，PRESET 命令を使えばよいでしょう．もう一つ，

```
PRESET(30,4),0
```

驚いたことに，PRESET 命令にも，色指定が付けられるのです．何だ，それでは，PSET 命令と同じではないかと思われた方もいることでしょう．確かに，その通りです．唯一，違う点は，色指定を省略した場合には，最後に実行した COLOR 命令の 2 番目のパラメータの値が使われます．どうやら，わかってきたでしょう．画面に，四角い部分が表われる前に実行したのは，

```
COLOR 7,2,0,7
```

でした．2 番目のパラメータは，2 です．2 は赤色に対応していましたね．このパラメータは，バックグラウンドカラーの設定を行います．バックグラウンドカラーとは，見ての通り，画面の地の色です．もちろん，まわりのふち取りの部分は，除きます．したがって，PRESET 命令を実行すると，現在のバックグラウンドカラーは，何色なのかを判断して，その色と同じ色で，ドットを表示してくれる訳です．実際は，それが，ドットを消しているように見えるだけなのです．

以上，ここまでを COLOR 命令を使って，まとめてみると，

```
COLOR 7,6,5,4
```

と書くと，１番目のパラメータは，７ですから，テキストの文字の色が，カラーモードであれば，白になります．白黒モードでは，リバースシークレットと呼ばれる白黒が反転して文字が表示されない状態になります．(やらない方がよいでしょう．)

２番目のパラメータは，６ですから，PRESET命令の色指定を省略すると，この６が使われることになります．６は，黄色でした．また，先程のCOLOR　７，６，５，４を実行して，グラフィックス画面をクリアすると，バックグラウンドカラーとして，黄色の四角い枠が現われます．

３番目のパラメータは，５ですから，画面の枠取りの部分は，水色となります．５は，水色でした．

４番目のパラメータは，４ですから，PSET 等のグラフィック命令 (PRESET 命令は除く) の色指定を省略した時には，この４が使われることになります．４は，緑色でしたね．

どんな画面になるのか，よくわからないのであれば，実際に実行してみましょう．

さて，最後になりますが，少し複雑な事を説明しなければなりません．読んでみて，わからなければ，無理に覚えることはないのですが，PC－8801 のグラフィックスを使いこなすのには，覚えなければならないことです．これから，PC－8801 のグラフィックスをたくさん使っていこうという方は，注意して読んでみて下さい．

今まで，１は青に対応していますね．とか，２は赤ですね．等と書いてきましたが，これらは，いつでも当てはまるということにならないのです．今回は，あなたをずいぶん，いらいらさせて，いろいろな設定をしてから，やり始めたので，問題は生じなかったのです．ですから，今まで，赤だとか青だとかの色を指定する時，この本に書かれている通りにやれば，その色が出てきたはずです．しかし，実際には，合計８色のカラーとその対応する数字（０～７）は，あなたが選ぶことができるのです．つまり，１に，あなたが，白を指定すれば，COLOR，１と２番目のパラメータを１にすると，バックグラウンドカラーは，青ではなく，白になるのです．わかりませんか．ちょっと，わかりづらい所です．また，今まで，２は赤と対応すると書いてきましたが，あなたが，２を青に指定すれば，COLOR ，２とすると，バックグラウンドカラーは，青になるのです．どうやら，２種類のカラーが出てきてしまって，わかりづらいようですから，新しい言葉を，

1つ出します．パレットというのがそれです．このパレットというのを，あなたが自由に指定できるのです．つまり，あなたは，0から7までの，8種類のパレットと呼ばれるものを持っています．そのパレットには，今まで使ってきた合計8色のカラーを，それぞれ自由に割りつけることができます．その割りつけを行う時には，今まで使っていた，カラー番号を使います．ですから，カラー番号の0は黒ですし，1は青ですね．あなたが覚えている通りです．次に，PC－8801の電源を入れた状態の時には，このカラー番号と，あなたが自由に色を割りつけることのできるパレット番号とは，全く同じです．ですから，パレット番号の0には，当然黒（カラー番号0）が割りつけられていますし，パレット番号の1には，青（カラー番号1）が割りつけられています．パレットと呼ばれるものが，よくわからなければ，パレット番号と，カラー番号は全く同じものと考えてさしつかえありません．しかし，一度パレットについて，わかってしまうと，かなり便利な事に気づかれるでしょう．なぜならグラフィックス命令のほとんどは，このパレット番号を使って，色指定をしているからです．（例外については，後から触れます．）そこで，一度パレット番号の0に，カラー番号の7つまり白を割りつけて，PSET命令で，色指定を0にしてドットを打っておき，今度は，同じパレット番号の0に，カラー番号の1つまり青を割りつけると，何と，先程白で打ったドットは，青に変わってしまうのです．わかりましたか．実際にやってみたい方もいるでしょうから，パレット番号にカラー番号を割りつける方法を書いておきます．

COLOR＝（パレット番号，カラー番号）

カラー番号 0：黒
〃　1：青
〃　2：赤
〃　3：紫
〃　4：緑
〃　5：水色
〃　6：黄色
〃　7：白

カラー番号も，一緒にのせておきますが，これは前に出てきたのと同じですから，誤解のないように．カラー番号自体は，絶対に変わりませんね．変わるのは，パレット番号の方ですよ．グラフィックス命令の中で，パレット番号で色指定をしない例外について，書いておきます．どの命令が例外なのかというと，何度も登場している，COLOR命令の，1番目と3番目のパラメータがそうです．もう何を

指定するパラメータなのかは，わかりますね．この2つのパラメータについては，パレット番号ではなくて，いつでもカラー番号を指定します．お間違いのないように．

パレット番号について，実際にいろいろ変えて試してみたいのですが，パレットについて，よくわからなかった方に対しては，たいくつな時間になるでしょうから，敢えてやらずにおきます．パレットについて，わかった方は，言われるまでもなく，すでに何かしら実行しているのではありませんか．パレットが合計8個もあると，しまいには，どのパレットに何色を割りつけているのか，わからなくなってしまいます．ですから，もし，パレットをいじった方は，次へ進む前に，次のことを実行しておいて下さい．

```
FOR I=0 TO 7:COLOR=(I,I):NEXT
Ok
```

何をしているのかは，もうおわかりですね．パレットを，元の状態に戻している訳です．

## 点をつなげると…

グラフィックス画面上に，ドットだけ打っていたのでは，あまりおもしろくないのですが，もう少しつき合って下さい．今度は，点をつなげてみます．もちろん，点をつなげれば，線になりますね．本当にそうでしょうか．やってみましょう．

```
FOR I=0 TO 639:PSET(I,99):NEXT
Ok
```

画面の中央に，直線が引かれましたか．

```
FOR I=0 TO 639:FOR J=0 TO 199:PSET(I,J):
NEXT J,I
Ok
```

今度は，斜めに引きましたか．確かにドットをつなげると，線になりますが，FOR～NEXT のループが出てきたりして，ちょっといやですね．それに，時間がかかるようです．もう少し便利な方法がないのかというと，ありました．LINE 命令です．

## もっと簡単に

線を引くには，LINE（ライン）命令．実際に，やってみましょう．先程書いた線を消します．

```
CLS 2
Ok
LINE(0,90)-(639,90),7
```

先程と同じ事が，FOR～NEXT 命令などを使わずに，わずか 2 行でできました．書き方は，今さら，説明することもないでしょう．上に書いた様なスタイルで，前のカッコの中に始点を，後ろのカッコの中に終点を指定すれば，始点から終点まで線を引いてくれます．その後ろにある色指定の 7 は，パレット番号ですが，パレットについて，よくわかっていなければ，カラー番号の 7 つまり，白で線を引かせていると考えて下さい．これで，短時間に，線を引くことができるようになりました．そうすると，一つ欲が出てきます．たとえば，折れ線グラフ等を画面に表示させる時に，2 本 3 本とその折れ線が多くなると，それぞれの線のスタイルを実線とか破線とか点線とか変えてみたくなります．これが，PC－8801 には，簡単にできてしまうのです．一度，画面をクリアします．

```
CLS 3
LINE(0,0)-(639,199),7
LINE(0,90)-(639,90),7,,&HFFFF
LINE(0,100)-(639,100),7,,&HF0F0
LINE(0,110)-(639,110),7,,&HAAAA
```

どうです，見事でしょう．どうしたら，描けるのかというと，上の LINE 命令をよく見て下さい．先程の LINE 命令に比べて，コンマが 2 つとその後ろに16進数で指定があります．コンマが 2 つあるということは，コンマとコンマの間に，一つパラメータが省略されていることがわかりますね．このパラメータについては，次で触れます．その次の& H で始まるのは，16進数です．この16進数の 4 桁の数字で，ラインスタイルつまり，実線とか破線とか点線とかを決めているのです．詳しい説明は，リファレンス・マニュアルの方を見て下さい．簡単に説明すると，この16進数の 4 桁を使うと，16個の点を横に並べた時に，16個の内，どの点とどの点を表示させようかという指定ができます．ですから，LINE 命令で線を引いた時には，その長さが16ドット分より長い場合には，16ドット分の線の繰り返しとなります．一度に指定できる長さは，最高16ドットということになります．後は，いろいろラインスタイルを変えて，試してみて下さい．かなり面白い線を引くことができますよ．

## ボックス（箱）を描く

先程省略した LINE 命令のパラメータには，B(BOX)または，BF(BOXFILL)のどちらかを書くことができます．もちろん，先程のように，全く省略してしまってもかまわない訳です．このパラメータに BF を指定すると，ラインスタイルの指定はできなくなります．BF というのは，何もボーイフレンドの略ではなく，ボ

ックスフィルの略です．ボックスフィルというのは，ボックス（箱）を描いて，そのボックスの内部をうめなさいという指定なのです．したがって，このパラメータにBを指定すると，ボックスを描いてくれることがわかりますね．やってみましょう．一度，画面をクリアして，

```
CLS 3
LINE(0,50)-(100,150),5,B
```

Bを指定した時には，ラインスタイルの指定もできます．省略した時には，&HFFFFつまり実線が使われます．始点と終点を指定するだけで，その2点を対角線とするボックスが，簡単に描けます．間違っても，斜めに傾むいた，平行四辺形などを，このLINE命令1つで書こうとはしないように，できませんよ．常に，二点を対角線とする四角形，正方形の場合もあれば，長方形の場合もありますが，斜めに傾むいた平行四辺形の場合はありません．

```
LINE(110,50)-(210,150),5,B,&HAAAA
LINE(220,50)-(320,150),5,BF
LINE(330,50)-(430,150),5,BF
```

同じように，始点と終点の二点を対角線としたボックスを描いた後に，その内部をうめるだけです．ラインスタイルの指定はできないことは，前に書いた通りです．書いたらどうなるかって，それはあなたが試してみればわかりますよ．

## 円を描くには

いよいよ，お待たせしました．この章の初めに，6種類の円をテキスト画面に，強引に描いてみましたが，これから，グラフィックス画面に，円を描いてみましょう．円を描くには，PSET命令を使って，ループを作ればよいのですが，またFOR～NEXTと，あなたに言われそうなので，円を書く命令，CIRCLE(サークル）命令を，すぐに紹介しましょう．CIRCLEと6文字あります．しっかり覚えて下さい．まずは，画面をクリアします．

```
CLS 3
Ok
```

次に，

```
CIRCLE(319,99),50,7
Ok
```

とうとう，やりました．円が描けましたね．えっ，大きさを変えてみたいって．どうぞ，カーソルを50の所へもっていって，値を変えてみて下さい．この50が半径ということになります．どうです．いろいろな大きさの円が描けるでしょう．余り半径を大きくすると，画面からはみ出してしまいますよ．えっ，もうはみ出

してるって．エラーメッセージは，返ってきませんから，どんどんはみ出しましょう．ちなみに，半径1の円とは，どんな円を描いてくれましたか．

CIRCLE命令の説明は，するまでもないように思えますが，実はちがいます．上の書き方では，パラメータを3つも省略しているのです．パラメータを省略しないで書いてみると，

```
CIRCLE(319,99),50,7,0,6.28318,.5
```

と，なります．後ろの3つのパラメータが，それぞれ省略された時には，これらの値が使われることがわかりますね．1つ1つ説明していきましょう．まず，CIRCLEに続けて，カッコでくくってあるのが，中心の位置の指定です．これもわかったと思います．次が，先程も言った様に，半径です．その次は，パレット番号です．わからなければ，カラー番号と考えて下さい．その次は，0となっていますが，これは，円を描き始める角度（開始角度）です．その次が，円を描き終わる角度（終了角度）です．最後が，比率です．後ろ3つのパラメータ，それぞれ，開始角度，終了角度，比率についての説明は，リファレンス・マニュアルの方に任せます．円を描いているだけならば，この後ろの3つのパラメータは省略してしまってかまいませんが，円孤を描こうとする時には，開始角度，終了角度が，問題になってくるでしょう．また，楕円を描きたい時には，最後の比率が問題になってきます．必要があれば，リファレンス・マニュアルを見て，覚えていって下さい．1つ面白い使い方としては，開始角度，終了角度にマイナスの値を使うと，中心からの半径が描かれ，扇形を描くことができます．使ってみると面白いでしょう．1つ，サンプルプログラムをのせておきます．興味のある方は，RUNしてみて下さい．

```
NEW
Ok
10 REM SAMPLE PROGRAM
20 CLS 3
30 PI=3.14159
40 FOR TH=0 TO 2*PI STEP PI/8
50    CIRCLE(319,99),150,5,-.001,-TH+.001
60    FOR I=1 TO 500
70    NEXT
80 NEXT
90 END
```

## 塗りつぶす

LINE命令を使って，ボックスを描いた場合，BFを指定すれば，そのボックスの内部を，塗りつぶしてくれました．同じように，円を描いた時に，その円の内

部を，塗りつぶしたくなりませんか．その時には，どうしたらよいのかというと，便利な命令，PAINT（ペイント）命令があります．1点を指定すると，その回りを指定した色でぬりつぶしてくれます．たとえば，CIRCLE命令を使って，パレット番号（わからなければカラー番号）7で円を描きます．次にこのPAINT命令で，円の内側を，パレット番号（わからなければカラー番号）5でぬりつぶしたい時には，

```
PAINT（中心位置），5，7
```

と書きます．つまり，後ろの3つのパラメータで，中心位置と領域色と境界色の指定をする訳です．やってみましょう．

```
CIRCLE(319,99),50,7
Ok
PAINT(319,99),5,7
```

見事に，円の内部が塗りつぶされましたか．敢えて，水色でとは言いませんが，パレットが元の状態であれば，円が白で描かれて，その内部が水色でぬりつぶされるはずです．PAINTに続けて，一点を指定する訳ですが，この点は，境界の内側の点であれば，どこでもかまいません．ただし，境界と重なっていては，いけません．ですから，先程の円の内部をぬりつぶすのに，別に（319，99）を指定しなくてもよいのですが，円の内部をぬりつぶすのであれば，その円の中心の位置をPAINTで指定しておけば，安全でしょう．

さて，領域色と，境界色を省略した時には，どうなるでしょう．この場合は，少々複雑で，領域色が省略されると，COLOR命令の4番目のパラメータで指定したフォアグラウンドカラーとなります．これは，PSET命令やその他のグラフィックス命令と同じですね．では，境界色を省略した時には，どうなるのかというと，これは，領域色と同じ指定のパレット番号（カラー番号）となります．したがって，両方とも省略すると，領域色，境界色のいずれも，フォアグラウンドカラーの指定と同じになる訳です．やはり，あまり省略せずに，しっかりと指定してからPAINTさせた方がよいでしょう．さもないと，思いもかけない所をPAINTされてしまうことになるかもしれません．気をつけましょう．また，PAINTしている際中に，もし，PAINTを中断させたいのであれば，STOPキーを押します．PC-8801はPAINTの途中で，きっとやめてくれるはずです．一画面をすべてPAINTさせるのには，かなり，時間がかかります．あなたの考えを裏切って，一画面をすべて，PAINTし始めてしまったのなら，迷わず，STOPキーです．

## 絵が動く

グラフィックス命令の，絵を動かしてみましょう．絵を動かすといっても，むずかしいことはありません．さっそく入力して下さい．

```
NEW
OK
100 CONSOLE 0,25,0,1:WIDTH 80,25:COLOR 7
:CLS
110 X=RND*76:Y=RND*23:XS=X:YS=Y:XA=1:YA=
1:SUM=0:CP=32:CH$="O"
120 LOCATE XS,YS:PRINT CH$;
130 IF X<1 THEN XA=1:BEEP 1
140 IF X>76 THEN XA=-1:BEEP 1
150 IF Y<1 THEN YA=1:BEEP 1
160 IF Y>22 THEN YA=-1:BEEP 1
170 X=X+XA:Y=Y+YA
180 IF PEEK(&HF3C7+X+Y*120)=CP THEN SUM=
SUM+1
190 LOCATE X,Y:PRINT "●";
200 XS=X:YS=Y
210 IF SUM<884 GOTO 120
220 LOCATE XS,YS:PRINT CH$;
230 SUM=0
240 IF CH$="O" THEN CH$=" ":CP=237 ELSE
CH$="O":CP=32:COLOR RND*6+1:CLS
250 GOTO 120
```

110行，190行，240行に○と●がありますが，これは，グラフィックキャラクタの白丸と黒丸です．今までのプログラムより，少しばかり長めですので，間違えないように．さて，では実行します．

```
RUN
```

黒丸が一つ画面上を動きます．そして動いた跡には，白丸が残されていきます．画面が，白丸で一杯になると，今度は，白丸を消していきます．そして画面上の白丸をすべて消してしまうと，最初に戻ります．つまり，ループしているわけです．プログラムの実行を止めるには，STOP キーを使って下さい．しばらくは，この動きをお楽しみ下さい．

プログラムの説明をしていきます．まず，このプログラムのように，絵を動かす場合，気を付けなければいけないことがあります．それは，先読みしておくことです．

何を先読みしておくのかというと，絵を動かす先の場所の状態の先読みです．

つまり，このプログラムでいえば，黒丸を一つ先へ進める時に，その一つ先の状態を調べて，把握しておき，黒丸がそこを通り過ぎた後に，元の状態に戻しておかなければいけません．このプログラムでは，180行で，この先読みをしています．が本来の意味とはちがう意味で，先読みをしています．

もう一つ大切なことは，絵の位置を覚えていなければなりません．絵を一つ先へ動かすには，今まであった絵を消さなければいけませんね．でないと，同じ絵がつながって表示されてしまいます．そして，この絵を消す時には，先程の先読みした状態を考慮しなければいけません．絵が通過した後を，元通りにしておくわけです．絵の位置として，現在，実際に絵のある位置と，一つ前に絵のあった位置とを，覚えているわけです．プログラムでは，それぞれ，X, Y と XS, YS にはいっています．

そして，新しい位置に絵を書く前に，一つ前にあった位置に，白丸あるいはスペースを書いています．実際に絵を動かすには，この X, Y を増加あるいは減少させればよいのですが，一つ注意があります．画面上からはずれると，エラーメッセージのお世話になります．そこで，140行から170行でその判断をしています．そして，X, Y にそれぞれ，XA, YA を足していますが，この XA, YA は値としては，－1か1をとるわけです．この XA が，1の時には，下向きに，－1の時には上向きに，また，YA が1の時には，右向きに，－1の時には左向きに動くことになります．そして，プログラムでは，その新しい場所の状態を先読みした後で，実際にそこへ黒丸を書き込みます．

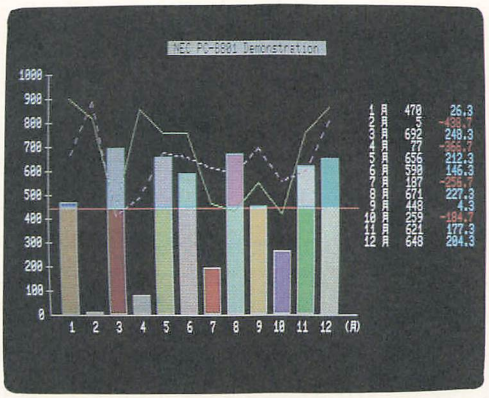
次に，今度は，そこの位置が一つ前の位置になるわけですから，X, Y をそれぞれ XS, YS に代入して，繰り返します．ループ脱出条件としては，白丸が画面一杯になるか，あるいは，白丸が画面からすべて消えた場合です．プログラムでは，210行がそうです．220行から240行までは，画面を白丸でうめるか，スペースでうめるかの選択をしているだけです．絵を動かすときには，一つのパターンがあります．そのパターンを覚えてしまえば，後は自由に絵を動かせるようになります．このプログラムから，そのパターンを握んでください．

## ビジネスに使うと　（円グラフ，棒グラフ）

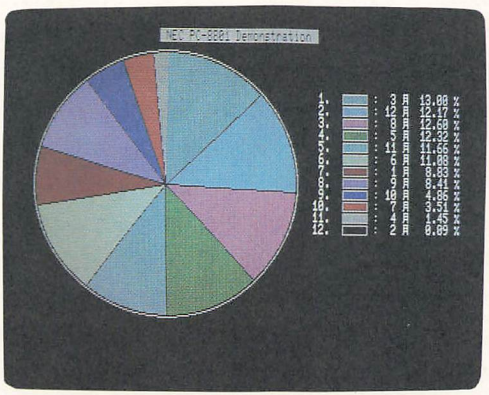
あなたが，PC－8801 をビジネスに使ってやろうと考えているのでしたら，これからやることは，多いに楽しんでもらえると思います．

今回，このために，円グラフと棒グラフを表示するプログラムを作りました．できれば，本書を読むのと，平行して，そのプログラムを，PC－8801 に実行していてもらいたいのです．カセットあるいはフロッピーディスクに，円グラフと棒

グラフを表示する，プログラムが入っていると思いますので，PC－8801 に LOAD して，RUN しておきましょう．一応，デモンストレーションをかねていますので，何もキーボードから入力しなくても，一度 RUN してしまえば，棒グラフと円グラフを交互に表示し続けます．




棒グラフ




円グラフ

プログラムについての説明は，省略します．興味のある方は，どうぞ，プログラムを読んでみて下さい．それ程，かわりにくいプログラムではないはずです．これまで覚えてきた事を使って，また，知らない命令は，リファレンス・マニュアル等で調べながら，プログラムの解析といきましょう．

では，何をしているプログラムかというと，１月から12月までのデータの処理をしています．まず，データとしては，０から9999までのデータを乱数により，12月分発生させます．次にそのデータから，平均を求め，それぞれのデータを，棒グラフとして表示させます．その時，赤線が平均を表わしています．(現在，PC－8801に棒グラフが表示されているのであれば，CTRL－Sにより，画面を一時停止させておくとよいのでしょう．)１月から12月までの乱数で作ったデータを，それぞれ，平均以下と平均以上とで，色分けしてあります．このように，PC－8801では，簡単に中間色を出せるのも特筆すべき点です．右側には，１月から12月までのソースデータと，平均以上であれば，水色で，どのくらい平均を上まわっているか表示し，平均に満たなければ，赤色で，どのぐらい平均を下まわっているのかを表示しています．二本の析れ線グラフが出ていますが，これは，ソースデータとは，一切関係ありません．乱数により，この析れ線も作っています．ラインスタイルには，&HFFFFと&HF0F0を使っています．このような使い方も，今までは，簡単にはできなかった事ですね．では，画面を止めて見ている方は，何かしらキーを押して，次の画面へ進みましょう．しばらく，棒グラフを表示した後に，今度は，円グラフの作成に入るでしょう．円グラフがすべて，表示された所を見計って，再度，CTRL－Sで画面を一時停止させておきましょう．この円グラフは，先程の棒グラフの時と同じデータを使って，今度は，各月の全体にしめる割合を大きい順に表示させています．見てもらえば，わかると思います．ただ，円グラフの回りに，もう一度円を描き終わった瞬間に，パレットを変更しています．具体的には，パレット番号の０から５までは，カラー番号のそれと同じですが，パレット番号の６には，白（カラー番号７）を割りつけ，パレット番号７には，黒（カラー番号０）を割りつけています．したがって，画面上から黄色がなくなり，黒が２つできたことになります．黄色がなくなったことから，ちょっとカラーの表現力が乏しくなりますが，PC－8801では，こんな色も出せるということを見ていただきたかったのです．さて，画面を止めて見ている方は，再び，動かしてみましょう．円グラフをしばらく表示すると，少し時間をおいてから，また，棒グラフを描き始めます．後は，今までと同じ繰り返しとなりますが，実は，１から12月までのデータをあなたが自由に入力することも可能です．その時には，棒グラフを書き終わって，その表示が消されるまでの間に，スペースキーを一回押します．すると，すぐに画面の表示が消されて，

```
Graph name : ｳﾘｱｹﾞ ｽｲｲ

 1.(January)          ?  1000
 2.(February)         ?  1250
```

```
 3.(March)          ?  1235
 4.(April)          ?  1450
 5.(May)            ?  1640
 6.(June)           ?  1780
 7.(July)           ?  1920
 8.(August)         ?  2010
 9.(September)      ?  1580
10.(October)        ?  1560
11.(November)       ?  1660
12.(December        ?  1680
```

と，聞いてきますから，グラフに付けたい名前を何か入力します．最大で，77文字まで可能です．すると，次に，1月から12月までのデータを聞いてきますので，それぞれ入れていきますが，許されるデータとしては，0から9999までとなっています．12月まで入れ終わると，棒グラフの作成にかかります．先程と同じように，棒グラフが表示されて，しばらくすると，次の円グラフの表示をするために，棒グラフは消されてしまいますので，必要であれば，CTRL－Sを使って，画面を止めておくとよいでしょう．棒グラフが，もうよいのであれば，何かしらキーを入力して画面を動かします．次に今，入力したデータの円グラフが表示されますから，必要があれば，画面を止めておきます．円グラフも，もうよいのであれば，何かしらキーを入力します．乱数による棒グラフ，円グラフの作成に入りますが，もし，プログラムを終了したい時には，円グラフの表示が消されだした時に，何かしらキーを押します．すると，画面をクリアした後に，ホームポジションにOkが表示されるはずです．プログラム中で，パレットを変えているために，できれば，STOPキー等を使って，プログラムを停めない方がよいと思います．プログラムを終了させたい時には，先程の手順に従って下さい．もし，STOPキーまたは，CTRL－Cでプログラムの実行を止めてしまった人は，次のダイレクトモードを実行してください．

```
FOR I=1 TO 7:COLOR=(I,I):NEXT
```

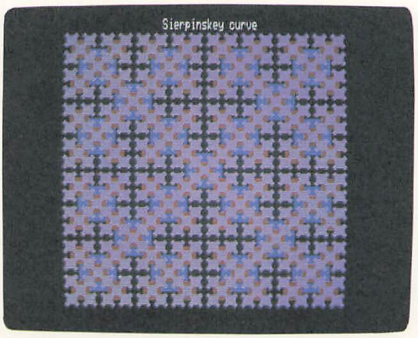
これで，パレットは初期状態と同じになります．

## 美しい図形を描くには

ここまで，PC－8801で使えるグラフィックス命令の説明をしてきました．もちろん，その全てを説明した訳ではありません．まだまだ，説明の足りない所もあります．特に，PC－8801では，今までにない概念を使って，グラフィックスを実現しています．これについては，一切触れていないのです．というのは，この新しい概念だけを説明するのでさえ，かなりの時間を必要としますし，すでに，リ

ファレンス・マニュアル，ユーザース・マニュアルの方で，これについては，詳しい説明がなされています．これから，PC－8801のグラフィックスを使ってみようという方は，ぜひとも，そちらをご覧下さい．PC－8801のグラフィックスとは，この第4章に書いたことだけではないのです．今まで別のコンピュータを使って，グラフィックスをやってこられた方も，PC－8801のグラフィックスには，その全てに驚かれるはずです．特に，縦が400ドット(640×400専用の高解像度モニタが必要です．)のモードにおいて，円を描いた時のその感激は，忘れられません．ぜひ，あなたも，その感激を味わってみて下さい．美しい図形を描くのには，まず，PC－8801のグラフィックス命令を正確に覚えることです．そしてまた，カラー番号0～7までの合計8色のカラーに加えて，中間色が簡単に出せるので，色の組合せが問題になってくるでしょう．これは，あなたのセンスの問題かもしれませんが……．

とにかく，PC－8801のグラフィックスの世界は，無限に広がっています．それを，うまく利用するのも，しないのも，あなたの手にかかっているのです．






## まとめ

以上，第4章では，テキスト画面に，円を描くことから始まり，実際に，グラフィックス画面に円を描いてみることもやりました．少しは，PC－8801のグラフィックスについて，わかってもらえましたか？パレットについては，どうでしょう．パレットという言葉を聞いただけでも，頭が痛くなりますか．まあ，そういわずに．棒グラフは，いかがだったでしょうか．自分なら，こうするのにと思った方は，どんどんプログラムを変更，改良してみましょう．それも，プログラミング上達のよい方法です．もし，自分の気に入ったプログラムができたのなら，カセットテープや，ディスケットに残しておきましょう．

それでは，次は第5章，最後の章です．がんばって読み進んで下さい．

# 第5章

# ストリングと配列

## あなたならどうする

もしあなたが学校の先生で，コンピュータを使って，テスト結果の平均点や標準偏差などを求めたいときはどうしますか？仮に生徒が40人いたとします．１人１人に少年Ａなどと変数を決めていきますか？それもよいでしょう．しかし平均点を求める式がとても長くなってしまいます．こんなとき配列変数が役に立ちます．配列変数とは普通の変数名の後に（　）を付けます．配列変数の実際の使い方を次に示しましょう．

```
A(5)=1:I=10:A(I)=3
```

カッコの中を配列変数の添字といいます．また添字には変数や式なども置くことができます．では配列変数を使って，５個のデータの平均を取るプログラムを作ってみましょう．

```
10 FOR L=1 TO 5
20  INPUT "DATA=";A(L)
30 NEXT L
40 FOR L=1 TO 5
50  H=H+A(L)
60 NEXT L
70 PRINT "AVERAGE=";H/5
```

これを配列変数を使わないで作ってみると，次のようになります．

```
10 INPUT "DATA=";A
20 INPUT "DATA=";B
30 INPUT "DATA=";C
40 INPUT "DATA=";D
50 INPUT "DATA=";E
60 H=(A+B+C+D+E)/5
70 PRINT "AVERAGE=";H
```

これはデータの数が５個なのであまり差は出ませんが，10個，20個と増えるにつれて差は歴然としてきます．よくわからない人は10個の平均を求めるプログラムを作ってみてください．

このように配列変数は，１つの変数名で多くの数値を記憶でき，かつ読み出せるのです．

## テーブルの上に並べると…

配列の考え方がちょっとよくわからないという人のために説明します．たとえばテーブルの上に５個のコーヒーカップが並んでいると想像してください．

実際に並らべても結構です．それぞれに角砂糖を数個入れてみてください．こ

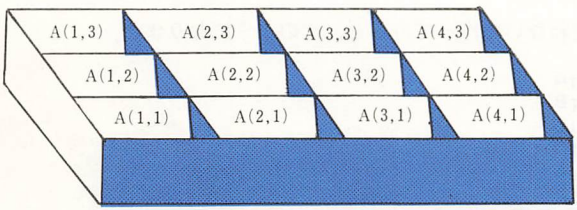



変数(１)～A(５)　配列変数A(n)

れを前節のプログラムにあてはめると，５個のコーヒーカップが変数A～E，角砂糖の数がデータです．

これでは平均を求めるときに全部の器を調べなければなりません．これに対して，配列の考え方は，１つの長い箱に５つの部屋があって，その部屋１つ１つに角砂糖が数個入っていると思えばよいのです．これならば箱を持ってくれば，５つのデータ全部が調べられます．

また配列の添字は，２つ並べて書くこともできます．これを２次元の配列と呼び，より多くのデータが扱えるようになります．この考え方を図に示します．




２次元配列A(m,n)

この２次元配列をプログラム中で使うにあたって，注意しなくてはならない事があります．それは，A（１，２）とA（２，１）は別の変数だということです．これは添字が定数のときはよいのですが，変数が入る時に充分注意しなくてはなりません．またこの２次元配列を使うことによって，行列の考え方をすることができます．

## 使わないと損をするDIM命令

ここでは配列を用いるにあたって注意しなくてはならないことをお話します．

まず第１に配列を使うにあたって本当は配列を宣言しなくてはならないという事です．宣言の仕方は次の通りです．

DIM 配列変数名

DIM は DIMension の略で，大きさという意味を持っています．配列変数名は，使用する配列の名前と添字の最大値を入れます．また配列変数名は，コンマで区切って並べて書くこともできます．たとえば配列変数名AとBとCを100個宣言するときは，次のようにします．

```
DIM A(100),B(100),C(100)
```

これでA (0) ～A (100)，B (0) ～B (100) C (0) ～C (100) までのエリアがメモリ上に取られました．が，しかし$N_{88}$(N)- BASIC では，配列は特に宣言しなくても，プログラムの中で配列変数が現れた時に自動的に10の大きさの配列が宣言されたことになります．ですから10以上の大きさの配列を使うときはきちんと宣言をしなくてはなりません．

```
20 FOR I=0 TO 11
30   A(I)=I
40 NEXT I
RUN
?BS Error in 30
Ok
10 DIM A(11)
RUN
Ok
```

第2に添字の最小値は0だということです．しかしこれは，OPTION BASE 文を使って変えることができます．つまり普通に配列を宣言すると添字は0から始まりますから，添字が1～最大値までしか変化しない場合は，使われない配列変数がでてきます．これを防ぐには0から変化させるか，OPTION BASE 文で添字の最小値を1にするのです．OPTION BASE 文の書式は次の通りです．

```
OPTION BASE 1 または 0
```

OPTION BASE 文は DIM 文の前に宣言しなくてはなりません．

第3に配列の添字が10以下の場合です．前の方に配列の自動宣言のことを書きましたが，添字の最大値が3くらいの場合です．BASIC は数値を表すのに4バイト使いますが，数値のエリアが11個分×4＝44バイト取られてそのうち使うのは3×4＝12バイトで，残り32バイトは無駄になります．これが3次元配列になると11×11×11×4＝5324バイトも取られてしまいます．ですから小さな配列も使うだけしっかり宣言しましょう．

## 同じ変数名，コンピュータはどう区別する

これまで配列変数について話してきましたが，単純変数との違いを少し説明しましょう．次の２つを見てください．

A５　　単純変数
A(５)　配列変数

この２つの変数は，どちらもAの後に５を付けて変数名としていますが，どちらも別々のメモリに値を記憶しています．次のことをやってみてください．

```
A5=3:A(5)=7
?A5,A(5)
 3             7
```

どうですか？それぞれ違う値を持たせてみるとよくわかると思います．

それともう１つ大きな違いがあります．それは，配列変数は間接指定ができるということです．たとえば，

```
I=5
AI=10:A(I)=12
? A5,A(5),AI
 3             12            10
```

というようにA５のつもりがAIに値を入れたらコンピュータは別のメモリに値をしまってしまいました．

しかし，A（I）の方にはしっかり値がしまわれています．つまり配列変数では，（　）の中の値が同じならば，同じ変数として処理してくれるのです．（もちろん添字のつく変数が違えば，添字が同じでも区別してくれます．例：A（５）とB（５）など）この間接指定ができるということは大変重要なことです．統計データの処理などでは，ループの中で繰返してデータの取り込みができるのです．たとえば次のプログラムを実行してみてください．

```
100 OPTION BASE 1:DIM A(5,5)
110 FOR I=1 TO 5
120  FOR J=1 TO 5
130   READ A(I,J)
140  NEXT J
150 NEXT I
160 FOR I=1 TO 5
170  FOR J=1 TO 5
180   PRINT " ";MID$("X+・",A(I,J),1);
190  NEXT J
```

```
200 PRINT
210 NEXT I
220 DATA 3,3,3,3,3
230 DATA 3,2,2,2,3
240 DATA 3,2,1,2,3
250 DATA 3,2,2,2,3
260 DATA 3,3,3,3,3
```

## 配列のエラーメッセージ，その傾向と対策

### ?BS Error (Bad subscript)

このエラーは，「添字が扱える範囲を越えた」という意味を持っています．

チェックポイント：添字がDIM文で宣言した大きさをこえていないか？
添字の個数（次元）はあっているか？

対策：添字をプリントアウトしてみる．
同じ配列変数名で異なる次元の配列変数を呼んでいないか？

例：?A (12)
?BS Error
?A (10, 10)
?BS Error

### ?DD Error(Duplicate Definition)

このエラーは，配列の二重宣言という意味を持っています．

チェックポイント：DIM文がループの中に入って何回も実行されていないか？
DIM文の後にOPTION BASE文がきていないか？
DIM文で宣言する前に配列変数を呼んで自動宣言されていないか？
同じ変数名で配列を宣言していないか？

対策：配列変数を数多く使用するときには，確認の意味をこめて，プログラムの先頭でまとめて宣言する．OPTION BASE文で添字の最小値を1とするときは，DIM文の前で宣言する．

### ?OM Error (Out of memory)

このエラーは，特に配列関係のエラーではありませんが，配列が大きすぎてメモリにエリアが取り切れないときに発生します．ここでは原因が配列に起因する

場合について書きます．

チェックポイント：配列が自動宣言されてメモリをむだ使いされていないか？
配列が大きすぎないか？

対策：配列を小さくする．
実数型配列を整数型配列に変える．

## 文字列（ストリング）を使う

あなたは，PC-8801に文字を表示させたいとき，どうしますか？もちろんPRINT文を使います．しかしPRINT文中の〝(ダブルクォーテーション）に囲まれた文字しか出力されません．これでは，変数の内容で出力する文章を変化させたいときなどに，IF文とPRINT文がプログラム中に数多く必要になってしまいます．またコンピュータに文字を入力，または記憶させたいときはどうしますか？これはもうお手上げではないでしょうか？ここで登場するのが，ストリング変数です．

このストリング変数はコンピュータに文字を入力したり記憶させたりできる他に，記憶した文字列をデータとして扱えるために，文字列の並べかえや切り離し，つけ足し，比較などが容易にできるのです．また先の変数の内容により出力する文章を変化させたいときなどは，配列変数の文字型を使うことによって容易に実現できます．

```
10 FOR LOOP=0 TO 10
20  READ NUMBER$(LOOP)
30 NEXT LOOP
40 INPUT "ENTER NUMBER (0-10)";N
50 IF N<0 OR N>10 THEN 40
60 PRINT N;"= ";NUMBER$(N)
70 DATA ZERO,ONE,TWO,THREE,FOUR,FIVE
80 DATA SIX,SEVEN,EIGHT,NINE,TEN
90 GOTO 40
```

このように行番号60だけで，11個の文字列を打ち分けることができます．この他に文字列の入力や比較ができることから，PC-8801を使ってデータベースなどを作ることが可能になります．それでは簡単な電話帳のプログラムを作ってみましょう．

このプログラムは，DATA文として名前と電話番号を持っています．これをREAD文を使って配列に取り込みます．このときに名前の場所に〝END〟という文字列があれば，そこでデータの取り込みを止めます．このプログラムの使い方

ですが，名前を登録した通りに入力し，〝ALL〟と入力すれば全ての名前と電話番号を出力します．

```
100 CLEAR:OPTION BASE 1:DIM TEL$(3,100),NAMAE$(100)
110 LOOP=1
120 READ NAMAE$(LOOP):IF NAMAE$(LOOP)="END" THEN 150
130 READ TEL$(1,LOOP),TEL$(2,LOOP),TEL$(3,LOOP)
140 LOOP=LOOP+1:GOTO 120
150 PRINT :PRINT "ﾄﾞﾅﾀ ﾉ ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ヲ ｵｻｶﾞｼﾃﾞｽｶ?"
160 LINE INPUT "ｵﾅﾏｴ ヲ ﾄﾞｳｿﾞ";WANT$:PRINT
170 IF WANT$="ALL" THEN 260
180 IF WANT$="" THEN 150
190 IF WANT$="END" THEN 250
200 '********** TEL No. SEARCH **********
210 LOOP=1
220  IF NAMAE$(LOOP)=WANT$ THEN GOSUB 310 :GOTO 150
230  IF NAMAE$(LOOP)="END" THEN 250
240 LOOP=LOOP+1:GOTO 220
250 PRINT WANT$;" ｻﾝ ﾊ ｷｵｸ ﾆ ｺﾞｻﾞｲﾏｾﾝ":GOTO 150
260 '********** PRINT ALL **********
270 LOOP=1
280  IF NAMAE$(LOOP)="END" THEN 150
290  GOSUB 310
300 LOOP=LOOP+1:GOTO 280
310 '********** PRINT TEL No. **********
320 TEL1$=RIGHT$("       "+TEL$(1,LOOP),6)+" -"
330 TEL2$=RIGHT$("    "+TEL$(2,LOOP),4)+" - "
340 PRINT NAMAE$(LOOP);TAB(16);"TEL ";
350 PRINT TEL1$;TEL2$;TEL$(3,LOOP)
360 RETURN
1000 'DATA ﾅﾏｴ,ｼｶﾞｲ ｷｮｸﾊﾞﾝ,ｼﾅｲ ｷｮｸﾊﾞﾝ,ﾊﾞﾝｺｳ
9999 DATA END
```

このプログラムはストリングを使ったほんの一例に過ぎません．上のプログラムに少し付け足しをすれば，住所を記憶させることもできますし数値変数と組み合わせることによって，在庫管理や売上伝票，請求書などの実務計算のプログラムを作ることもできるのです．また他にも記憶する項目をもっと増して自動検索機能を強力にすれば，顧客管理のプログラムを作ることもできます．このようにストリングを使うことによって，より実用的なプログラムの作成が可能となるのです．

## なぜストリングなのか？

前節で説明した様にリストリングはとても有効的です．しかしいくら有効だといっても使い方がわからなければどうしようもありません．そこでここでは実際の使い方を説明していきます．

まずストリング変数（文字型変数）です．これは普通の変数名の最後に＄（ダラ

ー）を付けることで文字型変数とすることができます．以下はストリング変数の書式例です．

| | |
|---|---|
| A$ | 文字型変数 |
| NEC$(2) | 文字型配列変数 |
| PC$(2, 4) | 上に同じ |

このようにBASICでゆるされる変数は，ストリング変数となることができます．しかし次の2つの文字変数は予約変数として常に値を持っています．

| | |
|---|---|
| DATE$ | 年月日を，YY／MM／DDの形で持っている．同じ形で代入可能． |
| TIME$ | 時分秒をHH：MM：SSの形で持っている．同じ形で代入可能． |

次に代入する方法ですが，まったく普通の変数の代入法と同じですが，次のことに注意しなければなりません．

1.左辺がストリング変数の場合，右辺も当然文字型にしなくてはなりません．

```
A$=10
?TM Error
```

2.右辺において文字列の演算は〝＋〟のみ認められています．やはり文字列を乗じたり，割ったり，引いたりはできないのです．

```
A$="123":B$="456":C$=A$*B$
?TM Error
```

3.1つの文字変数の中の文字数が255を越えてはいけません．

```
FOR L=1 TO 256 :A$=A$+"E":NEXT L
?SL error
```

この他にも，?OS Error（Out of string space）というエラーメッセージがありますがここでは触ずにおきます，くわしくはリファレンス・マニュアルを読んでください．以上の点に注意すれば，ストリング変数も普通の変数も変わりありません．次に代入式の例を示します．

```
  A$="ｺﾝﾆﾁﾊ"
NEC$="PERSONAL COMPUTER"
 PC$="PC-8801"
  N$=""
```

これらの代入文の右辺はみな文字定数（リテラル）と呼ばれ〝（ダブルクォーテーション）で囲まれた文字列が左辺のストリング変数に代入されます．しかし最後の代入文は〝（ダブルクォーテーション）が2つ並んでいるために文字列があり

ません．このような文字列はヌルストリングと呼ばれ何もデータを持っていません．またPRINT文で文字列を出力するときに使う，〝で囲まれた部分も文字定数なのです．

次にINPUT文中でのストリング変数について説明します．INPUT文中でも他の変数と同じように使うことができます．

```
10 INPUT "ENTER YOUR STRING ";A$
20 PRINT "YOUR STRING IS ";CHR$(34);A$;CHR$(34)
30 PRINT :GOTO 10
```

上のプログラムを打ちこんで実行してみてください．行番号20のストリング変数の両側のCHR$（34）は，あなたの打ちこんだストリングが，どのようにコンピュータに取り込まれたかわかるようにしたものです．CHR$の詳しい説明は，後でします．

```
ENTER YOUR STRING ? "HELLO                "
YOUR STRING IS "HELLO                "

ENTER YOUR STRING ? NEC,PC-8801
?Redo from start
ENTER YOUR STRING ?
```

このようにINPUT文におけるストリング変数の入力では，ストリングの前の空白は全て読み飛され後の空白は全て無視されます．意味のある空白を入力したいときは〝（ダブルクォーテーション）で囲めばよいのです．また，（コンマ）は変数の区切りのマークのために読み込まれません．しかしコンマやダブルクォーテーションを入力したいときは，どうしたらよいでしょうか？こんな時のためにLINE INPUTというステートメントが用意されています．LINE INPUTを使って先のプログラムを実行してみましょう．

```
10 LINE INPUT "ENTER YOUR STRING ";A$
20 PRINT "YOUR STRING IS (";A$;")"
30 PRINT :GOTO 10
ENTER YOUR STRING "HELLO                "
YOUR STRING IS ("HELLO                ")
ENTER YOUR STRING NEC,PC-8801
YOUR STRING IS (NEC,PC-8801)
```

このようにコンマやダブルクォーテーションなども入力できるようになります．しかしLINE INPUTでは，入力される変数は1つしか認められていません．10 LINE INPUT A$, B$などとはできないのです．ですからLINE INPUTとINPUTを場合によっては使い分ける必要が出てきます．たとえば多くのデータを多くの変数に入力する場合などです．

## 果してストリングはつながるか

ストリングの利点として，ストリングの連結があります．前節でストリングの演算は〝＋〞のみ認められている，と書きましたが，その結果はどのようになるのでしょうか．次のプログラムを入力してみてください．

```
10 P$="PC-8801"
20 INPUT "ｱﾅﾀ ﾉ ｵﾅﾏｴﾊ?",N$
30 A$=N$+" ﾄ "+P$+" ﾊ ﾄﾓﾀﾞﾁ!"
40 PRINT A$
RUN
ｱﾅﾀ ﾉ ｵﾅﾏｴﾊ?PC-6001
PC-6001 ﾄ PC-8801 ﾊ ﾄﾓﾀﾞﾁ!
```

このようにストリングの右端に，次のストリングの左端がピッタリとくっついて，新しい１つのストリングが出来上がります．

次にストリングの大小関係ですが，文字列を順次左から比較していってキャラクタコードの大きい方が大きいと判別されます．１文字目が同じときには２文字目でというように順番に比較していきますが，どちらかの文字列が終ってしまったときは，長い方の文字列が大きいと判別されます．こうして比較していき最後まで文字が同じときだけ，２つの文字列は等しくなるのです．次にあげる例を見て理解してください．

```
      "A" < "B"
     "AA" < "AB"
    "ABC" > "AB"
"STRING " < "STRINGS"
    "max" > "MIN"
   "1982" > "1981"
"PC-8801" = "PC-8801"
```

ストリングの比較で使える関係演算子は，＝，＜，＞，＜＞，＞＜，＜＝，＝＜，＞＝，＝＞で，数値の比較に使われるものと同じものです．

## ストリングは長さを持っている

あなたは次のストリングを見てどちらが長いと思いますか？

```
1) "PC-8801"
2) "ABCDEFGHIJKLMN"
```

もちろん2)の方が長いですね．しかし人間が目で見て長いというのはわかってもPC-8801にはわかってもらえませんし，文字数が必要なときもあります．こんなときにストリングの長さを求める関数があると便利です．それがLEN関数です．LEN関数の書式は次の通りです．

```
Y=LEN(X$)
```

X$は文字型の変数もしくは定数です．Yにはストリング X$の長さが入ります．ヌルストリングの場合は0が入ります．またストリングの中に出力されない文字（キャラクタコードで1～31までのもの）も数えられます．それでは，前節のプログラムに少し付け足してサンプルプログラムを実行してみましょう．

```
25 PRINT "STRING LENGTH =";LEN(A$)
LIST
10 LINE INPUT "ENTER YOUR STRING ";A$
20 PRINT "YOUR STRING IS (";A$;")"
25 PRINT "STRING LENGTH =";LEN(A$)
30 PRINT :GOTO 10
RUN
ENTER YOUR STRING NEC
YOUR STRING IS (NEC)
STRING LENGTH = 3
```

また出力されないキャラクタコードの例として，CTRL-Gをストリングとしてみましょう．これはダイレクトモードで実行してみます．

```
G$=CHR$(7)+CHR$(7)+CHR$(7)
? G$,LEN(G$)
                3
```

結果はどうなりましたか？ビープ音が3回して数字の3が出力されれば成功です．CHR$（7）を並らべれば並らべただけビープ音が出力され，数字が出ます．これでLEN関数の動作がおわかりいただけたと思います．これでもうあなたは，ストリングの長さを目で数える必要はなくなった訳です．

## キャラクタコード表を作る

あなたのPC-8801はあなたが文法誤りをしたときや，計算の結果がオーバーフローしたときや，メモリがなくなったとき，「ピッ」という音を発して知らせてくれます．この音はダイレクトモードのときにコントロールキーとGのキーを同時に押すことによっても発することができます．しかしプログラムの中で，ある処理が終ったら人間に知らせるためにこの，「ピッ」という音を使いたいと思うことがあります．

ではどうしたらこの音を発することができるでしょうか？次の通りに打ち込んでみてください．

```
PRINT ""
```

ダブルクォーテーションの中はCTRL-Gです．ではRUNさせてみてください．どうですか？何か聞えましたか？今のプログラムでは何も聞こえないはずです．何故ならコントロールコード（コントロールキーと共に押したアルファベット）は，キーボードからプログラムには入いらないからです．試しにプログラムをストリングの長さを打ち出すように変えて実行させてみてください．これはあなたにお任せします．ストリングの長さは0になるはずです．では次のプログラムを打ち込んで実行してみてください．

```
10 G$=CHR$(7)
20 PRINT "BEEP!";G$
RUN
BEEP!
```

どうですか？今度は何か聞こえたでしょう．このプログラムではG$の中に，あなたのPC-8801に音を出させる元，つまりCTRL-G（キャラクタコードで7）が入っているのです．このようにCHR$ (n) という関数は，nに対応するキャラクタコードを文字として与えてくれます．このCHR$関数を使うことによって，カーソル，コントロールキャラクタを出力することができるようになります．次の文を実行してみてください．

```
? CHR$(12)
```

どうですか？これはCLSと同じ働きをします．またこのCHR$関数はコントロールキャラクタだけではなく普通の文字も与えることができます．次のプログラムを実行してみてください．

```
10 INPUT "CHARACTOR CODE";C
20 IF C>255 THEN 10
30 PRINT "PRINT CHR$(";C;") = ";
40 PRINT CHR$(34);CHR$(C);CHR$(34):GOTO 10
```

それではPC-8801で表示できるキャラクタを出してみましょう．次のプログラムを打ち込んでください．キャラクタの上と左側に出ているのは，16進数です．16進のA～Fは，10進の10～15に対応しています．求めるキャラクタのコードは，左の数字を16倍して上の数字を足して求めます．

```
100 PRINT"   2  3  4  5  6  7  8  ";
110 PRINT"9  A  B  C  D  E  F"
120 FOR I=0 TO 15
130   PRINT HEX$(I);"  ";
140   FOR J=32 TO 255 STEP 16
```

```
150       PRINT CHR$(I+J);
160       PRINT "   ";
170     NEXT
180     PRINT
190 NEXT
200 END
```

## CHR$とASCは表と裏

前節では数字からキャラクタを作り出しました．ここでは，キャラクタから数字を作る関数を説明します．この関数をASC関数と言います．

```
ASC(X$)
```

すると，文字列X$の最初の文字のキャラクタコードを値とします．本来はキャラクタコードではなくASCIIコードを値としていたいのですが，PC-8801では，カタカナやグラフィック記号なども表示できるのでその値も返してきます．

```
10 LINE INPUT "X$=?";X$
20 IF X$="" THEN 10
30 PRINT "ASC(X$)=";ASC(X$)
40 PRINT "CHR$(ASC(X$))=";CHR$(34);CHR$(ASC(X$));CHR$(34)
50 PRINT :GOTO 10
RUN
X$=? ABCDE
ASC(X$)= 65
CHR$(ASC(X$))="A"
X$=? ♠
ASC(X$)= 232
CHR$(ASC(X$))="♠"
```

(ASCではnull（"）は使えません)

## 時間を切る

ここでは最も基本的な文字関数である，MID$，LEFT$，RIGHT$について説明します．まずLEFT$関数からです．

```
LEFT$(X$,I)
```

この関数は文字列X$の左側のI文字だけ取り出す関数です．次のプログラムを実行してみてください

```
10 INPUT "X$=";X$
20 FOR I=0 TO LEN(X$)
30   PRINT LEFT$(X$,I)
40 NEXT I
```

```
10 N$="123456789"
20 S$="          "
30 FOR I=0 TO 9
40   PRINT RIGHT$(S$+LEFT$(N$,I),9)
50 NEXT I
```

次にRIGHT$関数です.

```
RIGHT$(X$,I)
```

このRIGHT$関数は, LEFT$関数と逆の働きをします. つまり文字例X$の右側のI文字を取り出す関数です. またこの関数を使うことによって, 右詰めの文字列を作ることができます.

```
10 INPUT "X$=";X$
20 FOR I=0 TO LEN(X$)
30   PRINT RIGHT$(X$,I)
40 NEXT I
RUN
X$=PC-8801

1
01
801
8801
-8801
C-8801
PC-8801
```

この2つの関数によって, 文字列の右側や左側を取り出すことができます. しかし文字列のまん中を取り出すにはちょっと工夫が入りますね. でも安心してください. そのための関数が用意されています. それがMID$関数です.

```
MID$(X$,I,J)
```

このMID$関数は文字列X$のI番目の文字からJ個の文字を取り出す関数です. またJは省略することができ, そのときは文字列X$のI番目の文字から最後までの文字列を取り出します. また文字列X$の文字数が, Iより小さいときはヌルストリングとなります.

```
10 INPUT "X$ =";X$
20 FOR I=1 TO LEN(X$)
30    PRINT SPC(I-1);MID$(X$,I,3)
40 NEXT I
50 PRINT X$
```

```
X$ =? BASIC
BAS
 ASI
  SIC
   IC
    C
BASIC
```

それでは，MID$関数をRIGHT$関数とLEFT$関数で作ってみましょう．

```
LEFT$(RIGHT$(X$,LEN(X$)-I+1),J)
```

それでは，3つの関数の応用として年月日と時分秒を表示させるプログラムを作ってみましょう．

```
100 CLS
110 NEN$=LEFT$(DATE$,2)
120 TUKI$=MID$(DATE$,4,2)
130 HI$=RIGHT$(DATE$,2)
140 JI$=LEFT$(TIME$,2)
150 HUN$=MID$(TIME$,4,2)
160 BYO$=RIGHT$(TIME$,2)
170 YEAR$="19"+NEN$+" 年 "+TUKI$+" 月 "+HI$+" 日 "
180 HOUR$=JI$+" 時 "+HUN$+" 分 "+BYO$+" 秒 "
190 HYOJI$=YEAR$+"       "+HOUR$
200 LOCATE 0,0:PRINT HYOJI$;
210 GOTO 110
```

## SPACE$はSPCを兼ねる？

PRINT文中で文字列と文字列の間にスペースをあけるために，SPCという命令があります．これを使ってプログラムを作ってみました．

```
10 FOR I=1 TO 3
20   PRINT "NEC";SPC(I);"PC-8801"
30 NEXT I
RUN
NEC PC-8801
NEC  PC-8801
NEC   PC-8801
```

このようにSPC (n) という命令はn個のスペースを出力します．しかしSPACE$ (n) という関数を使っても同じことができます．この関数はn個のスペースを持つストリングを与えます．次のプログラムを実行してみてください．

```
10 FOR I=3 TO 1 STEP -1
20   PRINT "NEC";SPACE$(I);"PC-8801"
30 NEXT I
RUN
NEC   PC-8801
NEC  PC-8801
NEC PC-8801
```

どうですか？前のプログラムとの違いがわかりましたか？どちらも指定した数だけのスペースを出力します．しかしSPACE$関数は代入文でも，PRINT文中でも使えますが，SPC関数はPRINT文中とPRINT #文中でしか使えません．またSPACE$関数を使って，グラフを書くこともできます．次のプログラムを実行し

てみてください．

```
10 WIDTH 80
20 FOR L=0 TO 10
30  X=L*(10-L)*2
40  PRINT SPACE$(X);"*"
50 NEXT L

RUN
*
                    *
                              *
                                        *
                                             *
                                               *
                                             *
                                        *
                              *
                    *
*
```

また BASIC では文字変数などは初期化されるとヌルストリングになるので，ストリングを足していって目的のストリングを作る訳ですが，ストリングを右詰め出力したい時に，最初にスペースだけの文字列を作っておき，スペースとストリングを結合して，RIGHT$関数で右側だけを取り出して出力するのに使えます．

## 便利な命令STRING$

STRING$関数は，キャラクタが指定した数だけ並んだストリングを与える関数です．書式は次の通りです．

```
STRING$(N,C)
STRING$(N,X$)
```

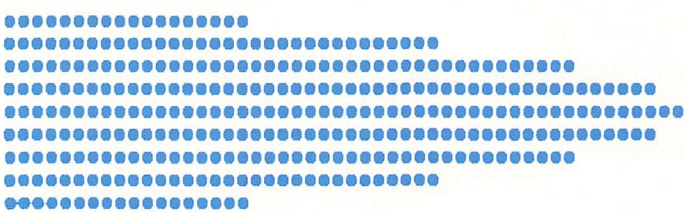
前者はキャラクタコードの文字を，n 個並べた文字列を与えます．後者は X$の最初の文字を n 個並べます．また前節の SPACE$関数は，STRING$関数で表すと次のようになります．

```
SPACE$(N) <=> STRING$(N," ")
```

では前節のグラフのプログラムに STRING$関数を使ってみましょう．

```
10 WIDTH 80
20 FOR L=0 TO 10
30  X=L*(10-L)*2
40  PRINT STRING$(X,236)
50 NEXT L
```

```
RUN
```



## 似ているようでも全然違う

前節のSTRING$関数に似ている関数に，STR$関数がありますが，働きは全く違います．STR$関数の書式は次の通りです．

```
STR$(X)
```

STR$関数はXの数値を文字列に変換します．

```
X=1234.56:? STR$(X),X
 1234.56         1234.56
```

ここで注意しなくてはならないことがあります．それはプリントアウトの形が同じだからといって，次のようなことはできません．

```
Y=STR$(X)
?TM Error
Ok
A=1000+STR$(X)
?TM Error
```

文字型と数値型を混同しないようにしてください．

## 数値とストリングの変換はこの命令で

前節では数値を文字列に変換することをしました．では文字列を数値に変換するには，どうしたらよいでしょうか．そのためのVAL関数が用意されています．そうでなくては世の中，不公平というものです．次にVAL関数を示します．

```
VAL(A$)
```

これは「文字列（数字列）→数値」の変換を行う関数です．

```
X$="123"
PRINT VAL(X$)
 123
```

を実行してください．見かけはただ，

```
PRINT X$
```

とした場合と変わりがありません．しかし，

```
X$=" 123"
PRINT VAL(X$)+123
 246
```

を実行することによってVAL(A$)は数値として扱われていることがわかるでしょう．このVALは数字列を数値に変換するため，数字列の始めが，+，−，&，数字以外のときには，値として0を返してきます．

```
? VAL("123")    123
? VAL("-123")  -123
? VAL("ABC")    0
? VAL("-ABC")   0
```

このVALを使えば，16進数などのINPUTが，容易になります．

```
100 INPUT "HEX$= ";H$
110 H=VAL("&H"+H$)
```

また，2度目の指数部表現のEやD，小数点が表れると，それ以降に数字があっても無視されます．

```
? VAL("12.34.56")              12.34
? VAL("12E3E4")                12000
```

## 数値と比べると

STR$とVAL関数で数値と文字列の相互変換が可能となりました．しかし数値と文字列では根本的に違います．それは当然ですが，ここでは問題とするのはPRINT文中においての事です．次の2つを比べてみてください．

```
? 1234
 1234
? "1234"
1234
```

PRINT文で数値を出力すると数字の前に符号が入るスペースが開けられ，数字の後にもスペースが1つ置かれます．これに対して文字列は，いきなり文字が始まります．これについては次のことを実行してみてくだされば一目瞭然です．

```
Q$="###":A=1234:A$="1234"
? Q$;A;Q$,Q$;A$;Q$
### 1234 ###  ###1234###
```

このように数値の前後には自動的にスペースが置かれます．それではSTR$関数を使って，数値を文字列に変換するとどうなるのでしょうか？　実際にやってみましょう．

```
? Q$;A;Q$,Q$;STR$(A);Q$
### 1234 ###  ### 1234###
```

数値を文字列に変換すると，後のスペースがなくなります．しかし，数字の前に符号が入るためのスペースが開いていますね．もしあなたがこのスペースも取りたいと思ったのならば，

```
MID$(STR$(X),2)
```

とすればよいのです．これはMID$関数の第2引数を省略した形です．次のプログラムを実行してみてください．3つの出力の違いがわかると思います．

```
10 INPUT "X=";X
20 A$=STR$(X)
30 B$=MID$(A$,2)
40 Q$="#"
50 PRINT"  X","STR$(X)","MID$(STR$(X),2)"
60 PRINT Q$;X;Q$,Q$;A$;Q$,Q$;B$;Q$
RUN
X= 12.3456
  X             STR$(X)         MID$(STR$(X),2)
# 12.3456 #    # 12.3456#      #12.3456#
```

## ストリングをサーチする

あるストリング変数の中に，特定の文字または文字列が入っているのかいないのか，また，それは始めの文字からいくつめにあるのかを調べるには，どうしたらいいのでしょう？下記のプログラムを見てください．

```
100 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
110 INPUT "STRING ";N$
120 L=LEN(N$)
130 IF L>26 OR L=0 THEN 110
140 FOR I=1 TO 27-L
150  IF MID$(A$,I,L)=N$ THEN 180
160 NEXT I
170 PRINT "NOT FOUND":GOTO 110
180 PRINT N$;" No.";I
190 PRINT :PRINT :GOTO 110
```

このプログラムは，INPUT されたSTRINGが，アルファベットの中のどこに存在するか，又は存在しないか，そして存在する場合は，いくつめの文字から始まっているのかを出力するものです．

このように，ストリング変数中の文字列を探がす場合には，ループを使うという方法があることがわかったでしょう．しかし，コンピュータには，このループを使わなくても，ストリング変数中の特定のストリングを探し出す命令があります．それは，INSTRです．

前述のプログラムをINSTR命令を使って書きかえてみます．

```
100 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
110 INPUT "STRING ";N$
120 N=INSTR(A$,N$)
130 IF N=0 THEN PRINT "NOT FOUND":GOTO 110
140 PRINT N$;" No.";N:PRINT :PRINT :GOTO 110
```

どうですか，大分簡単になりました．さらにINSTR命令では，ストリング変数の始めから何番目以後において，特定のストリングの有無を発見したいときのために，

```
INSTR(N,X$,Y$)
```

という型が用意されています．これは，X$のN番目の文字以降に，Y$があるかないかを調べるものです．INSTR（X$，Y$）と同様に，X$中にY$がある場合は，そのY$の始まる始めの文字の番号を，存在しないときには0を返してきます．

```
A$="ABCDEFGHIJ"
B$="DEF"
PRINT INSTR(3,A$,B$)
 4
```

尚，X$にY$が複数存在する場合は，1番初めのY$についてのみ値を返してきます．

このINSTR命令の応用して1つのストリング変数中に，いくつの特定ストリングが存在するのか，その個数を調べるプログラムを載せておきます．

```
100 A$="ABCDFIMNABOXZYABCABXYAB"
110 I=1:C=0
120 PRINT A$
130 N=INSTR(I,A$,"AB")
140 IF N=0 THEN END
150 PRINT SPC(N-1);"AB"
160 C=C+1:I=N+2
170 GOTO 130
```

## まとめ

この章では，配列と文字列について書いてきました．この2つのプログラムの要素は，実用的なプログラムには欠かせないものです．しかしその反面，気を付けなくてはならないことも数多く出てきます．

たとえば，不容易に配列を取っていくとメモリの無駄使いを仕兼ねないこと．また，データの保存の仕方などです．文字列の有用性について説明しているときに出てきた電話帳のプログラムでは，データをDATA文としてプログラム中に持っていたために，プログラム全体をロード，セーブしなくてはならないということです．これは，ディスクを使ったり，カセットテープにデータファイルとしてセーブしたりすることによって解決できます．やはり大容量，高速ということが要求されますと，フロッピーディスクの方がよいでしょう．フロッピーディスクにすることによって，データの付け足し，訂正，削除などが，簡単にできるようになります．

それから文字列の，入力についてですが，INPUT，LINE INPUT文では，入力ごとにキャリッジターンを押さなければなりません．1文字しか入力しないときや，入力する文字数が決まっているときなどは，INKEY$やINPUT$を使った方がよい時があります．次のプログラムを入力してください．

```
10 INPUT "HOW LONG IS YOUR STRING";L
20 PRINT "ENTER ";L;" CHARACTERS";
30 I$=INKEY$
40 IF I$="" THEN 30
50 C$=C$+I$:C=C+1
60 IF C<>L THEN 30
70 PRINT
80 PRINT "YOUR STRING =";CHR$(&H34);C$;CHR$(&H34)
```

ではRUNしてみてください．最初の何文字入れるか聞いてきますので，数字を入れてください．

```
HOW LONG IS YOUR STRING? 7
```

次に何文字打ち込んでくださいと聞いてきますから，指定された文字数を打ち込んでください．このとき打ち込んだ文字は，エコーバック（確認のために打ち込んだ文字を出力すること）されませんので，画面には出ません．ここでは，PC-8801と打ち込みます．

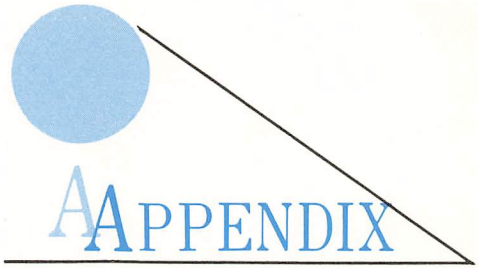
```
ENTER  7  CHARACTORS
```

指定された文字数だけを入力すると，

```
YOUR STRING ="PC-8801"
```

と出力されます．INPUT$は指定した文字数だけ入力する命令です．これは少し使い方が難しいですから，よく理解をしてください．INKEY$は，キーボードが押されていればその文字を，押されていなければヌルストリングを値とする関数のようなものです．また文字関数の所では，触れませんでしたが，文字関数の引数（STR$（X））は，0～255までの範囲になければなりません．小数点以下は，四捨五入されます．それから配列の添字は，255次元まで，値は0～65535までに納まっていなければなりません．負の値のときは，？FC Error（Illegal function call）エラーになります．

しかし恐れてばかりいたのでは，何も生まれません．PC-8801の可能性は，あなた自身の創造力によって無限に広がっているのです．




A. スクリーンエディタPart II

B. 時計, 日付プログラム

C. キャラクタセット表

D. PC-8001 ソフトウェアを利用するには

APPENDIX A

# スクリーンエディタ (PartII)

いよいよ，スクリーンエディタのPart IIです．スクリーンエディタを使う上で，知っていると便利なことをいくつか紹介します．大きく分けて，2つのことをこれからやっていきます．1つは，ファンクションキーについてです．もう1つは，第1章でも登場したコントロールキーを伴なったキーの使い方についてです．さっそく，ファンクションキーからやっていくことにしましょう．すでに，現在ファンクションキーに定義されている内容を知る方法を，あなたは知っていますね．もちろん，CONSOLE命令の3番目のパラメータで，ファンクションキーの表示が画面の最下行に出ます．SHIFTキーを押していなければ，f・1からf・5までの内容が反転して表示されますね．そして，SHIFTキーを押すことにより，今度は，f・6からf・10までの内容がわかります．しかし，気が付いた方もいると思いますが，これでは1つのファンクションキーに定義されている内容がすべて表示されるとは，限りません．そこで，定義されている内容をすべて知りたい時にはどうしたらよいのかというと，便利な命令があります．

```
key list

load "          save "
auto            key
go to           print
list            edit .
run             cont
```

10個分，表示されましたね．これが，現在ファンクションキーの1から10までに定義されている内容です．そして，PC-8801の電源を入れた時や，リセットスイッチを押した時には，常に，このようにファンクションキーは定義されている訳です．(あなたがすでに，ファンクションキーの定義の内容を変えていなければの話しですが……．)では，ファンクションキーの内容を定義するにはどうしたらよいでしょう．例として，現在，f・5に入っている内容と同じものをf・1に定義

してみます.

```
KEY 1,"RUN"+CHR$(13)
```

どうなりましたか.

f・1とf・5が全く同じ内容になりましたか. これで, ファンクションキーの定義が自由にできるようになりましたね. 1つだけ, CHR$(13)というのがよくわからないかもしれませんが, これは ↵ (リターンキー) を押すのと同じ働きをするのです. APPENDIXに, キャラクタセット表が載っていますので, その表の13の所を見ると, $^{C}_{R}$ と書かれているはずです. この $^{C}_{R}$ というのが, ↵ と同じである訳です. その他にもすべてのキャラクタがこのCHR$で表わせますから, 先程の変わりに,

```
KEY 1, CHR$(&H72)+CHR$(&H75)+CHR$(&H6E)+CHR$(13)
```

先程と同じ内容が定義されましたね. しかし文字の場合には, "(ダブルクォーテーションマーク) で囲めば済むのですから, それに従った方がよいでしょう. また, 1つのファンクションキーには, 最大15文字分定義できますから, 何か便利な使い方を定義しておきましょう. はじめに触れたように, 画面の最下行に表示させた時にはこの15文字の頭から, 40桁の場合には5文字, 80桁の場合には11文字だけ表示されます.

それでは次に, コントロールキーを併ったキーの説明に入りましょう. 第1章で, 合計18個ある中の10個についてはやりました. 残りの8個についてそれぞれ見ていきますが, 言葉で説明するよりも実際にあなたにやってみてもらった方がよいと思いますので, 画面にリストが表示できる状態にしてください. 具体的には何かのプログラムをPC-8801にロードして, リストを画面に表示させてください. すでにプログラムがPC-8801に入っている方は, わざわざロードする事はありません. とにかく最低10行ぐらいのリストを画面に表示させて, カーソルをホームポジションに戻します.

```
CTRL-F
```

カーソルが次のワードに進みましたか. ワードという言葉がわかりませんか. その質問には, PC-8801が答えてくれます. CTRL-Fによって, カーソルの動いた部分が, 1ワードということになります. CTRL-Fにより, 1つ先のワードへカーソルを進ませることができます.

CTRL - B

今度は,CTRL-F の逆で，1つ前のワードへ戻りましたね.CTRL-F とCTRL-B は2つをひとまとめにして覚えておきましょう．もう1つ，カーソルを速く動かせるのは，

CTRL - X

です．これは，現在カーソルのある行の最後に表示されている文字の次へ，カーソルを動かしてくれます．カーソルを速く動かしたい時には，役に立つでしょう．

カーソルを，行の最初に持っていきます．そして，

CTRL - E

一瞬にして現在カーソルのある行がなくなりました．今度はカーソルを，次の行の最初ではなくて，1つ右へ動かします．カーソルの左側には何かしら文字が表示されている状態にしておいて，

CTRL - E

です．わかりましたか．CTRL-Eは現在カーソルのある位置以降にあるその行を削除してくれます．これと似ているのには，

CTRL - U

があります．こちらは，カーソルの位置にかかわらず，現在カーソルのある行を全て削除します．さらにもう1つ，ワード単位で削除する場合には，

CTRL - D

を使います．どこからどこまでが1ワードかは，PC - 8801 が教えてくれることは，前にも書きましたね.

さて，以上6個出てきましたので，残りは2個となりました.

CTRL - J

次は CTRL - J ですが，これは，INS キーと併用して使います．つまり，1行を2行に分けたい場合がプログラミングしていると，よく出てきます．その時にはその分割したい行の2行目にしたい頭の文字へカーソルを持っていき，INS キーを SHIFT キーを押さえながら押します．この状態で何文字でも挿入すること

ができる訳ですが，ここで，

CTRL - J

と押します．とにかくやってみてください．1度やってみるとどんな働きをするのか，よくわかると思います．

CTRL - O

最後に，CTRL - O が残りました．次のプログラムを入力して，どんな働きをするのか見てみましょう．

```
100 FOR I=1 TO 500
110 PRINT USING "#####";I
120 NEXT
130 END
```

RUN してみるのですが，CTRL - O をいつでも押せる準備をしておいて，RUN してみます．画面には I の値が 1 から500まで，合計500個表示されるはずなのですが，CTRL - O を 2 秒おきぐらいに何度か押してみます．何と，CTRL - O を 1 回目に押すと，次に CTRL - O が押されるまで，I の表示はされませんね．何度かやって確かめてください．そして，CTRL - S のように，プログラムの実行を止めている訳ではなく，2 回目に CTRL - O を押した時には，I の値は，最後に表示された I の値と連続していませんね．実はこの CTRL - O は，押す毎に ON, OFF が繰り返されるトグルスイッチになっています．ON の状態とは，プログラム中にある PRINT 文は，通常通り実行されます．つまり，普通の状態です．OFF の状態とは，プログラムの PRINT 文は，全て無視されて，プログラムの実行はそのまま続いています．その証拠に，プログラムを RUN した後ですぐに CTRL - O を 1 回押すと，それ以降 I の表示はされませんが，しばらくするとプログラムの実行は終了します．試してみるとよいでしょう．これはかなり有効な使い方ができると思います．大いに活用しましょう．

これで全ての説明が終わりました．この Part II では，知っていると便利なことを書いてきましたが，知らなくてもそれほど困ることはないと思います．ですから，少しずつ，あなたの頭のメモリの許せる範囲で覚えていけばよいでしょう．

それでは，第 1 章とこの Part II で説明した全てのコントロールコードの一覧表をのせておきますので，活用してください．

N88-BASICのコントロールコード表

| キャラクタコード | 対応するキー | 機　　能 |
|---|---|---|
| 1 | CTRL-A | ヘルプキーと同じ (N88) |
| 2 | CTRL-B | 1つ前のワードへ戻る |
| 3 | CTRL-C | 実行の停止．ストップ・キーと同じ |
| 4 | CTRL-D | 1ワード削除 (N88) |
| 5 | CTRL-E | 1行削除 |
| 6 | CTRL-F | 1つ先のワードへ進む (N88) |
| 7 | CTRL-G | スピーカを鳴らす |
| 8 | CTRL-H | バックスペース |
| 9 | CTRL-I | 水平タブ．タブキーと同じ |
| 10 | CTRL-J | ラインフィード，インサートモードで2行に分割 |
| 11 | CTRL-K | ホームポジション |
| 12 | CTRL-L | テキスト画面クリア |
| 13 | CTRL-M | キャリッジリターン |
| 14 | CTRL-N | 1つ先のワードへ進む (N) |
| 15 | CTRL-O | 画面への表示を無効にする (N88) |
| 18 | CTRL-R | インサートモード |
| 19 | CTRL-S | 表示を一時停止する (N88) |
| 21 | CTRL-U | 1行キャンセル (N88) |
| 24 | CTRL-X | 1行の最後へ進む (N) |
| 27 | ESC | 表示を一時停止する (N) |
| 28 | → | カーソルを右へ移動 |
| 29 | ← | カーソルを左へ移動 |
| 30 | ↑ | カーソルを上へ移動 |
| 31 | ↓ | カーソルを下へ移動 |

APPENDIX B

# 時計,日付プログラム

```
100 ' CLOCK & DATE
110 CONSOLE 0,25,0,1:WIDTH 80,25:COLOR 7,0,0,7
120 SCREEN 0,0:WINDOW(0,0)-(639,199):VIEW(0,0)-(639,199)
130 DEFINT A-Z:DIM TIME!(2),SX!(2),SY!(2)
140 PI!=3.14159:PI2!=PI!*2:DA=0
150 FOR I=0 TO 2:SX!(I)=319:SY!(I)=99:NEXT
160 ON HELP GOSUB 430:HELP STOP:CLS 2
170 CIRCLE(319,99),210,7,,,.4:CIRCLE(319,99),300,7,,,.33
180 TILE0$=CHR$(&HF)+CHR$(&HFF)+CHR$(&HF0)
190 TILE1$=CHR$(&H0)+CHR$(&H0)+CHR$(&HFF)
200 BGS$=STRING$(3,&H0)
210 PAINT(50,99),TILE0$,7,BGS$:PAINT(0,0),TILE1$,7,BGS$
220 PAINT(0,0),TILE1$,7,BGS$:GOSUB 370:SCREEN ,0:HELP ON
230 IF DA=VAL(RIGHT$(DATE$,2)) GOTO 250
240 LOCATE 36,7:PRINT DATE$;:DA=VAL(RIGHT$(DATE$,2))
250 TIME!(0)=VAL(RIGHT$(TIME$,2))+45:SW=0:CL=7:K=30:GOSUB 330
260 LOCATE 36,17:PRINT TIME$;
270 TIME!(1)=VAL(MID$(TIME$,4,2))+45:SW=1:CL=4:K=28:GOSUB 330
280 TIME!(2)=((VAL(LEFT$(TIME$,2)) MOD 12)+(TIME!(1)-45)/60)*5+
45
290 SW=2:CL=3:K=20:GOSUB 330
300 CIRCLE(319,99),3,7:PAINT(319,99),7,7
310 IF TIME!(0)<>VAL(RIGHT$(TIME$,2))+45 GOTO 230 ELSE GOTO 310
320 '
330 X!=K*COS(PI2!*TIME!(SW)/60)*5+319:Y!=K*SIN(PI2!*TIME!(SW)/6
0)*2.5+99
340 IF (SX!(SW)<>X! OR SY!(SW)<>Y!) THEN LINE(319,99)-(SX!(SW),
SY!(SW)),0
350 LINE(319,99)-(X!,Y!),CL:SX!(SW)=X!:SY!(SW)=Y!:RETURN
360 '
370 LOCATE 38,3:PRINT "XII":LOCATE 30,4:PRINT "XI":LOCATE 49,4:
PRINT "I"
380 LOCATE 22,7:PRINT "X":LOCATE 57,7:PRINT "II":LOCATE 16,12:P
RINT "IX"
390 LOCATE 61,12:PRINT "III":LOCATE 21,17:PRINT "VIII"
400 LOCATE 57,17:PRINT "IV":LOCATE 29,20:PRINT "VII":LOCATE 49,
20:PRINT "V"
410 LOCATE 39,21:PRINT "VI":RETURN
420 '
430 HELP OFF:SCREEN ,2:CLS
440 PRINT "OLD DATE=";DATE$:PRINT "OLD TIME=";TIME$:PRINT
450 INPUT "NEW TIME=";IN$:IN=LEN(IN$):IF IN<>0 THEN TIME$=IN$:I
N$=""
460 INPUT "NEW DATE=";IN$:IN=LEN(IN$)
470 IF IN<>0 THEN DATE$=IN$:DA=VAL(RIGHT$(DATE$,2)):IN$=""
480 CLS:GOSUB 370:DA=0:SCREEN ,0:HELP ON:RETURN
```

APPENDIX C

# キャラクタセット表

上位4ビット→

| 下位4ビット↓ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | $D_E$ | | 0 | @ | P | | p | | | | ー | タ | ミ | | |
| 1 | $S_H$ | $D_1$ | ! | 1 | A | Q | a | q | | | 。 | ア | チ | ム | | 円 |
| 2 | $S_X$ | $D_2$ | " | 2 | B | R | b | r | | | 「 | イ | ツ | メ | | 年 |
| 3 | $E_X$ | $D_3$ | # | 3 | C | S | c | s | | | 」 | ウ | テ | モ | | 月 |
| 4 | $E_T$ | $D_4$ | $ | 4 | D | T | d | t | | | 、 | エ | ト | ヤ | | 日 |
| 5 | $E_Q$ | $N_K$ | % | 5 | E | U | e | u | | | ・ | オ | ナ | ユ | | 時 |
| 6 | $A_K$ | $S_N$ | & | 6 | F | V | f | v | | | ヲ | カ | ニ | ヨ | | 分 |
| 7 | $B_L$ | $E_B$ | ' | 7 | G | W | g | w | | | ァ | キ | ヌ | ラ | | 秒 |
| 8 | $B_S$ | $C_N$ | ( | 8 | H | X | h | x | | | ィ | ク | ネ | リ | ♠ | |
| 9 | $H_T$ | $E_M$ | ) | 9 | I | Y | i | y | | | ゥ | ケ | ノ | ル | ♥ | |
| A | $L_F$ | $S_B$ | * | : | J | Z | j | z | | | ェ | コ | ハ | レ | ◆ | |
| B | $H_M$ | $E_C$ | + | ; | K | [ | k | { | | | ォ | サ | ヒ | ロ | ♣ | |
| C | $C_L$ | → | , | < | L | ¥ | l | \| | | | ャ | シ | フ | ワ | ● | |
| D | $C_R$ | ← | − | = | M | ] | m | } | | | ュ | ス | ヘ | ン | ○ | |
| E | $S_O$ | ↑ | . | > | N | ^ | n | ~ | | | ョ | セ | ホ | ゛ | | |
| F | $S_I$ | ↓ | / | ? | O | _ | o | | | | ッ | ソ | マ | ゜ | | |

APPENDIX D

# PC-8001ソフトウェアを利用するには

## PC-8801のソフトウェアの利用について

PC - 8801 には $N_{88}$ - BASIC の他に姉妹機に採用されているものと同じ N - BASIC も搭載されています.

ですから，現在流通している，N - BASIC でプログラミングされたものであれば PC - 8801 上ですぐに走らせることができます.

## $N_{88}$-BASICからN-BASICへ

PC - 8801 に電源を入れると，最初に起動するのは，必ず $N_{88}$ - BASIC です.

$N_{88}$ - BASIC から N - BASIC への切り換えは，二つの方法があります.

1．PC - 8801 本体内部のディップスイッチ（ハードウェアスイッチ）によって切り換える.（ユーザーズ・マニュアル参照）

2．$N_{88}$ - BASIC の命令によって切り換える.
$N_{88}$ - BASIC によって，N - BASIC への切り換えができる NEW ON（ソフトウェアスイッチ）コマンドがあります．しかし，N - BASIC から $N_{88}$ - BASIC への切り換えはできません.
リセットを押すか，電源を入れ直す必要があります.

ディップスチッチによって切り換えを行った方は，リセットや電源の入れ直しでは，切り換わりませんので，ディップスイッチを元に戻してください.

## これだけは注意してください

$N_{88}$ - BASIC から N - BASIC へ切り換わると，今まで，$N_{88}$ - BASIC で走って

いたプログラムは，すべて消えてしまいます．

あらかじめ，カセットテープやディスクにセーブしておいてください．

$N_{88}$- BASIC と N - BASIC では，カセットとディスクに関する入出力命令が異なっていますので，リファレンス・マニュアルを良く読んでから，操作してください．

### 例：カセット

[$N_{88}$] LOAD "CAS1 : DEMO"
SAVE "CAS1 : DEMO"

[N] CLOAD "DEMO"
CSAVE "DEMO"

### 例：ディスク

[N] MOUNT1
SAVE "DEMO"
REMOVE1

[$N_{88}$] SAVE "DEMO"

$N_{88}$- BASIC の場合，MOUNT，REMOVE 命令はオートマチックマウント，リムーブによって BASIC が管理します．

NEC 日本電気株式会社・新日本電気株式会社

7811311

PC-8801 BASIC入門

NEC